

POWER DECODING REED–SOLOMON CODES UP TO THE JOHNSON RADIUS

JOHAN S. R. NIELSEN

Technical University of Denmark,
Department of Applied Mathematics and Computer Science
Denmark

(Communicated by the associate editor name)

ABSTRACT. Power decoding, or “decoding using virtual interleaving” is a technique for decoding Reed–Solomon codes up to the Sudan radius. Since the method’s inception, it has been an open question if it is possible to incorporate “multiplicities”, the parameter allowing the Guruswami–Sudan algorithm to decode up to the Johnson radius. In this paper we show that this can be done, and describe how to efficiently solve the resulting key equations. As the original Power decoding, the proposed algorithm is a one-pass algorithm, where decoding follows immediately from solving a shift-register type equation. It is a partial decoding and will fail to return a codeword for a few received words; we investigate its failure behaviour theoretically as well as giving simulation results. This is an extended version where we also show how the method can be made faster using the re-encoding technique or a syndrome formulation.

1. Introduction. Power decoding was originally proposed by Schmidt, Sidorenko and Bossert for low-rate Reed–Solomon codes (RS) [34]. Using shift-register synthesis techniques, the method can decode as many errors as the Sudan algorithm [40]. As opposed to Sudan’s list decoder, Power decoding is a one-pass algorithm where decoding is realised by solving a simultaneous shift-register problem; however, Power decoding always returns at most one codeword and will for a few received words simply fail. For random errors simulations indicate this occurs with only very small probability.

The Sudan decoder generalises to the Guruswami–Sudan decoder [14] by introducing the multiplicity parameter, improving the decoding radius for all rates and allowing it to decode up to the Johnson radius [15]. Since [34], it has been an open question whether it is likewise possible to introduce a “multiplicity parameter” into Power decoding and thereby increase the decoding radius up to the Johnson radius.

In this work we settle this question in the affirmative. The overall behaviour of the decoder is similar to Power decoding:

1. The equations are of a generalised shift-register type, and no root-finding as in Guruswami–Sudan is necessary.
2. The decoding radius becomes almost exactly that of the Guruswami–Sudan decoder (under the same choices of parameters).

2010 *Mathematics Subject Classification.* Primary: TODO TODO; Secondary: TODO.
Key words and phrases. TODO TODO TODO.

3. When a codeword is returned, it is always a closest codeword. The method will fail for a few error patterns whenever one decodes beyond half the minimum distance.

Furthermore, we will show how to realise the decoder efficiently, yielding a complexity $O^{\sim}(\ell^{\omega} sn)$, where ω is the exponent of matrix multiplication, and s, ℓ are the multiplicity, respectively powering parameters of the decoder, and $O^{\sim}(\cdot)$ is big- O but ignoring $\log(ns\ell)$ factors. This is very close to the best known complexities $O^{\sim}(\ell^{\omega-1} s^2 n)$ for the Guruswami–Sudan algorithm or the Wu list decoder [7].

We also investigate the failure behaviour of the proposed method. Though we do not settle the question of precisely how often Power decoding fails, we make some headway: we prove that the behaviour depends only on the error, and not the sent codeword, and we show that failure can only occur beyond half the minimum distance. We then give a closed upper bound on the probability for one choice of the algorithm’s parameters, $(s, \ell) = (2, 3)$. We also present simulation results that demonstrate a very small probability of failure for larger parameter choices.

Compared to the two existing list decoders, the Guruswami–Sudan and the Wu algorithm, we believe the proposed algorithm is interesting for several reasons. Firstly, it is more practical since the decoder needs only a single sub-algorithm—of shift-register type—while the other two decoders are more involved. The Guruswami–Sudan algorithm consists of two steps: interpolation and root-finding. Interpolation is comparable to the computation in Power decoding (see next section), but root-finding is an additional, non-trivial step. E.g. in [1], a hardware implementation of Köetter and Vardy’s soft-decision variant of Guruswami–Sudan has the critical path in the root-finding unit, and this also uses a significant area of the entire circuit. For hard-decision, a lower multiplicity would be more relevant, and then interpolation unit would be significantly simpler while the root-finding unit would not, causing the latter to occupy relatively more area and latency. Anecdotally, Broadcom also recently revealed interest in using joint decoding of Interleaved Reed–Solomon codes in a communication standard [12]; here it was crucial that the decoder was a one-pass shift-register-based algorithm, based on [35]: a method that is computationally very related to Power decoding.

Secondly, Power decoding has also been applied in settings where the list decoders have not been formulated, such as improved decoding of Interleaved RS codes [44], and in compressed sensing [22], among others [21, 24]. It is clear that the proposed extension of Power decoding immediately extends to decoding of Interleaved RS codes, but analysing the details is outside the scope of this paper. Examining this as well as the other applications are interesting avenues of future work.

Lastly, Power decoding decodes beyond half-the-minimum distance without the use of interpolation at all. This is a rarity in algebraic decoding, and it demonstrates that interpolation is not necessary for decoding up to the Johnson radius; at least in the presence of random errors. This sheds new light on decoding of RS codes, and represents an important puzzle piece in relation to the two list-decoding algorithms.

Parts of these results were presented at ACCT-14 [28].

1.1. Related Work. Power decoding was introduced in [36]: for low-rate RS codes, it was shown how one can compute generalised syndromes from “powering” the received word, and that these can be used for efficient decoding by solving a multi-sequence shift-register synthesis problem. One chooses a “powering degree” ℓ : higher ℓ yields better decoding radius, but is admissible only for lower-rate codes. In [36],

a bound on the failure probability was given for RS codes over binary extension fields when $\ell = 2$, but a general conjecture was given based on simulations results. The failure behaviour was then further examined in [47] and [27], where bounds on the failure probability were obtained over any field for $\ell = 2$ and $\ell = 3$. In [27], a reformulation of Power decoding was given based on Gao's decoder [10], and this was used to show that whether or not Power decoding fails depends only on the error pattern, and not the sent codeword. Power decoding has been applied in various other settings: for decoding Interleaved Reed–Solomon codes beyond the usual collaborative bound [44]; for improved decoding of Complex Reed–Solomon codes for use in Compressed Sensing [22]; decoding of Hermitian codes [24]; and for decoding CRT and Interleaved CRT codes [21].

The Guruswami–Sudan algorithm [14] is a polynomial-time list-decoding algorithm up to the Johnson radius $J_{n,k} = n - \sqrt{n(k-1)}$ [15]. “List-decoding” means that the algorithm will return *all* codewords within the decoding radius. For the algorithm one chooses two parameters $s, \ell \in \mathbb{Z}_+$, usually dubbed “the multiplicity” respectively “the list size”. They satisfy $s \leq \ell$, and they need to grow large for attaining the best decoding radius: for a decoding radius of $J_{n,k} - \varepsilon n$, one needs $s, \ell \in O(1/\varepsilon)$ for any $\varepsilon \in \mathbb{R}_+$. See [26, p. 58] for an extreme numerical example.

As noted already in [36], Power decoding is related to Guruswami–Sudan when $s = 1$ (also known as “Sudan decoding” after [40]): choosing the same value for ℓ yields (almost) exactly the same decoding radius. Computationally, there are more similarities, as noted below.

Guruswami–Sudan consists of two phases, usually dubbed “Interpolation” and “Root-finding”: first, one finds an “interpolation polynomial” $Q(y) \in \mathbb{F}[x][y]$, and then one finds $\mathbb{F}[x]$ -roots of it. Both phases have received a tremendous amount of attention with the aim of speeding them up, e.g. [2, 4, 7, 8, 19, 32, 46]; see [7] for an overview on the literature for the Interpolation step. The best currently known complexities are $O(\ell^{\omega-1} s^2 n)$ for Interpolation [7], and $O(\ell^2 s n)$ for Root finding, proposed in [2] with a tighter complexity analysis in [5]. Without the use of fast arithmetic, the best complexities are $O(\ell^3 s^2 n^2)$ for Interpolation [46], respectively $O(\ell^2 s^2 n^2)$ for Root finding [32].

One approach for fast Interpolation in Guruswami–Sudan has been to formulate “Interpolation key equations”, as in [32] for the case $s = 1$, and [46] for the general case. These are shift-register-type equations whose solution result in an interpolation polynomial. These are related to Power decoding: the generalised syndromes in [32] equal those of the original Power decoding [34]. However, the two sets of key equations are inherently *different*: the solution to the Power decoding equations yields the error locator, while no clear notion of an error locator is known for the Guruswami–Sudan algorithm. Similarly, the key equations that we derive in Section 3 bears a resemblance to the equations of [46], and it is an interesting question what the algebraic relation between the two approaches is.

Another avenue of research on fast Interpolation for Guruswami–Sudan is to apply $\mathbb{F}[x]$ -lattice basis reduction to an explicit constructible $\mathbb{F}[x]$ -matrix [2, 4, 8, 19]. This lattice depends on two polynomials, G and R (see page 5), which are also used in Gao's minimum-distance decoding algorithm [10], and which appear in both the Gao reformulation of Power decoding [27] and the algorithm we give here. Inspired by this line of research, lattice basis reduction was previously applied to solve the key equations of the original Power decoding [25], and we will solve the new Power decoding key equations by extending this approach.

The Wu decoding algorithm [45] is an amalgamation between classical key equation decoding [6] and the Guruswami–Sudan: one first attempts half-the-minimum distance decoding using the classical key equation (see the following section). If this fails, the polynomials computed in the failed attempt are then used to set up a problem solvable by an $\mathbb{F}(x)$ -variant of the Guruswami–Sudan algorithm. One again needs Interpolation and Root-finding sub-algorithms which are similar to but slightly more involved than for Guruswami–Sudan; see e.g. [5, 7, 42] for work on these. The best complexities for these steps equal those of the Guruswami–Sudan algorithm [5, 7]. However, from a practical perspective, the Wu algorithm is the more complicated to implement since one also needs classical key equation decoding, and since the $\mathbb{F}(x)$ -variants of the sub-algorithms are technically more involved. The Wu algorithm is also a list-decoding algorithm, and also decodes up to $J_{n,k}$. Also here one needs to choose parameters s, ℓ , whose growth relate to the decoding radius as in the Guruswami–Sudan algorithm [5].

1.2. Organisation. In Section 2 we give an introduction to the previous key equation-based decoding algorithms: half-the-minimum distance and Power decoding. In Section 3, we then derive the new key equations. These are non-linear relations between polynomials revealing the error, and we describe in Section 4 how to use them for obtaining an efficient decoding algorithm. We derive a decoding radius in Section 5, and relate it directly to that of the Guruswami–Sudan algorithm. Power decoding will fail on certain received words within this radius, however, and we investigate this in Section 6. In Section 7 we give simulation results. In Section 8 and Section 9 we investigate re-encoding respectively syndrome reformulations of the proposed key equations, providing practical – if not asymptotic – speedups to the decoder.

The decoding method has been implemented in Sage v6.9 [39] and can be downloaded from <http://jsrn.dk/code-for-articles>, together with the code for running the simulation.

2. Preliminaries and Existing Key Equations. In complexity discussions, we count arithmetic operations in the field \mathbb{F} . We will use ω as the exponent for matrix multiplication, i.e. $2 \leq \omega \leq 3$. We use $O^\sim(\cdot)$ as big- O but ignoring log-factors. In a few places we also use $M(n)$ to denote the complexity of multiplying together two polynomials of degree at most n ; we can trivially use $M(n) \in O(n^2)$ or we can have $M(n) \in O^\sim(n)$, see e.g. [43].

2.1. GRS codes. Consider some finite field \mathbb{F} . Choose $n \leq |\mathbb{F}|$ as well as distinct $\alpha_1, \dots, \alpha_n \in \mathbb{F}$ as well as non-zero (not necessarily distinct) $\beta_1, \dots, \beta_n \in \mathbb{F}$. For any $f \in \mathbb{F}[x]$ we write

$$\text{ev}(f) = (\beta_1 f(\alpha_1), \dots, \beta_n f(\alpha_n)) .$$

The $[n, k, d]$ Generalised Reed–Solomon (GRS) code is the set

$$\mathcal{C} = \{ \text{ev}(f) \mid f \in \mathbb{F}[x], \deg f < k \} .$$

The α_i are called *evaluation points* and the β_i *column multipliers*. \mathcal{C} has minimum distance $d = n - k + 1$ and the code is therefore MDS.

Consider now that some $\mathbf{c} = (c_1, \dots, c_n)$ was sent with $\mathbf{c} = \text{ev}(f)$ for some $f \in \mathbb{F}[x]$, and that $\mathbf{r} = (r_1, \dots, r_n) = \mathbf{c} + \mathbf{e}$ was the received word with error $\mathbf{e} = (e_1, \dots, e_n)$. Let $\mathcal{E} = \{i \mid e_i \neq 0\}$ and $\epsilon = |\mathcal{E}|$.

Note that column multipliers can be ignored in decoding: we simply compute $\mathbf{r}' = (r_1/\beta_1, \dots, r_n/\beta_n) = \mathbf{c}' + \mathbf{e}'$, where \mathbf{c}' is in the code \mathcal{C}' which has the same evaluation points α_i but where all $\beta_i = 1$. \mathbf{e}' is an error vector with the same number of errors as \mathbf{e} . In the remainder of the article, we therefore assume $\beta_i = 1$.

Introduce two essential polynomials, immediately computable by the receiver:

$$G = \prod_{i=1}^n (x - \alpha_i) \quad R : \deg R < n, \quad R(\alpha_i) = r_i, \quad i = 1, \dots, n.$$

G can be pre-computed, while R is computed upon receiving \mathbf{r} using Lagrange interpolation.

Key equation decoders revolve around the notion of an error locator Λ and error evaluator Ω :

$$\Lambda = \prod_{j \in \mathcal{E}} (x - \alpha_j) \quad \Omega = - \sum_{i \in \mathcal{E}} e_i \zeta_i \prod_{j \in \mathcal{E} \setminus \{i\}} (x - \alpha_j).$$

where $\zeta_i = \prod_{j \neq i} (\alpha_i - \alpha_j)^{-1}$. Note that $\epsilon = \deg \Lambda > \deg \Omega$.

The following relation between the four polynomials are is at the heart of our investigations:

Lemma 2.1. $\Lambda(f - R) = \Omega G$.

Proof. The closed formula for Lagrange interpolation implies that $f - R = \sum_{i=1}^n -e_i \zeta_i \prod_{j \neq i} (x - \alpha_j)$. This directly means

$$\Lambda(f - R) = \Lambda \sum_{i \in \mathcal{E}} -e_i \zeta_i \prod_{j \neq i} (x - \alpha_j) = \sum_{i \in \mathcal{E}} -e_i \zeta_i \left(\frac{\Lambda}{x - \alpha_i} \right) G = \Omega G.$$

□

The objects $\mathbf{c}, \mathbf{r}, \mathbf{e}, \Lambda$, etc. introduced here will be used in the remainder of the article.

2.2. Classical Key Equations. Let us revisit the key equation implicit in Gao's decoder [10], which follows directly from Lemma 2.1:

$$\Lambda R \equiv \Lambda f \pmod{G}. \quad (1)$$

This is a non-linear equation in the unknowns Λ and f , and it is not immediately obvious how to build an efficient decoder around it. The good idea is to *ignore* the non-linear relation: we replace the sought quantities Λ and Λf with unknowns λ and ψ , both in $\mathbb{F}[x]$, and such that

$$\lambda R \equiv \psi \pmod{G}.$$

This is now a linear relation, but unfortunately with infinitely many solutions. We further restrict the solutions by requiring

$$\deg \lambda + k - 1 \geq \deg \psi.$$

Note that this is satisfied if λ is replaced by Λ and ψ by Λf . Finally, we seek such λ, ψ where λ is monic and has minimal degree. The hope is now that $\lambda = \Lambda$ even though we solved for a much weaker relation than (1); effectively, it is therefore the low degree of $(\Lambda R \pmod{G})$ which is used to solve for Λ . Solving such requirements for λ and ψ is sometimes known as rational function reconstruction [43] or Padé approximation [3]. They are easy to solve for in complexity $O(n^2)$ or $O^\sim(n)$, using e.g. the extended Euclidean algorithm [9, 10, 41].

It can be shown that whenever $\epsilon < d/2$ we get $\lambda = \Lambda$ and $\psi = \Lambda f$, see e.g. [10]. Then $f = \psi/\lambda$ and decoding is finished. However, whenever $\epsilon \geq d/2$, the approach will essentially never work.

Whenever 0 is not an evaluation point, i.e. $\alpha_i \neq 0$ for all i , then the equation can be rewritten to the more classical *syndrome key equation* [6]. First some notation: for $p \in \mathbb{F}[x]$, let $\text{rev}(p, d)$ denote the *reversal of the coefficients* of p at degree d , i.e. $\text{rev}(p, d) = x^d p(x^{-1})$ for some integer $d \geq \deg p$. To lighten the notation, we will often omit the d -argument when there is an implied upper bound on the degree of the polynomial being reversed; to be precise, note that we then reverse on the *upper bound* on the degree, and not on the actual degree which might happen to be lower.

Introduce $S(x)$ as the power series expansion¹ of $\text{rev}(R)/\text{rev}(G)$ truncated at x^{n-k} . Then by reversing Lemma 2.1 at degree $\epsilon + n - 1$ we get:

$$\begin{aligned} \Lambda R &= \Lambda f - \Omega G && \Longleftrightarrow \\ \text{rev}(\Lambda R, \epsilon + n - 1) &= \text{rev}(\Lambda f, \epsilon + k - 1)x^{n-k} - \text{rev}(\Omega G, \epsilon + n - 1) && \implies \\ \text{rev}(\Lambda)\text{rev}(R) &\equiv -\text{rev}(\Omega)\text{rev}(G) \pmod{x^{n-k}}. \end{aligned}$$

Since $x \nmid \text{rev}(G)$ this implies the well-known formula:

$$\text{rev}(\Lambda)S \equiv -\text{rev}(\Omega) \pmod{x^{n-k}}. \quad (2)$$

A (now less obvious) algebraic relation exist between $\text{rev}(\Lambda)$ and $\text{rev}(\Omega)$. To allow for efficient solving, we forget this relation, and replace $\text{rev}(\Lambda)$ and $-\text{rev}(\Omega)$ by unknowns $\hat{\lambda}$ and $\hat{\omega}$, and solve for the minimal degree $\hat{\lambda}$ satisfying

$$\begin{aligned} \hat{\lambda}S &\equiv \hat{\omega} \pmod{x^{n-k}} \quad \text{and} \\ \deg \hat{\lambda} &> \deg \hat{\omega}. \end{aligned}$$

This time the modulus is a power of x ; solving such an equation for $\hat{\lambda}$ and $\hat{\omega}$ is known as Padé approximations [3] or a linear feedback shift-register synthesis [31, Section 6.7]. It can be solved in complexity $O(n^2)$ or $\tilde{O}(n)$ using either the extended Euclidean algorithm or the Berlekamp–Massey algorithm.

One can again show that this approach will succeed, i.e. in the end $\hat{\lambda} = \text{rev}(\Lambda)$, whenever $\epsilon \leq \lfloor (d-1)/2 \rfloor$ [6]. Slightly stronger, one can show that the approach will succeed if and only if the Gao key equation approach succeeds [27].

2.3. Simply Powered Key Equations. Power decoding, or decoding by virtual interleaving [36], is a generalisation of (1) where not one but multiple non-linear relations between Λ and f are identified, essentially still based on Lemma 2.1. The original formulation of [36] is based on the classical syndrome key equation, while powering the Gao key equation was described in [27]. We will begin with the latter:

Lemma 2.2 (Simply Powered key equations). *For any $t \in \mathbb{Z}_+$ then*

$$\Lambda R^t \equiv \Lambda f^t \pmod{G}.$$

Proof. By Lemma 2.1 we have

$$\Lambda f^t = \Lambda(R - (R - f))^t = \Lambda R^t + \Lambda(R - f)(\dots) \equiv \Lambda R^t \pmod{G}.$$

□

¹By inserting the explicit Lagrange interpolation formula for R , it is easy to see that this definition of the syndrome polynomial corresponds to the classical one, in e.g. [31, Section 6.2].

Again this gives non-linear relations between Λ and f . To solve them efficiently, we use only the first ℓ of the equations, for some chosen ℓ , and introduce unknowns $\lambda, \psi_1, \dots, \psi_\ell \in \mathbb{F}[x]$. We then solve for λ, ψ_t such that λ is monic and of minimal degree such that

$$\begin{aligned} \lambda R^t &\equiv \psi_t \pmod{G}, & t = 1, \dots, \ell \quad \text{and} \\ \deg \lambda &\geq \deg \psi_t - t(k-1). \end{aligned}$$

Finally, we hope that the found $\lambda = \Lambda$. In that case $f = \psi_1/\lambda$ and decoding is finished.

Note that there is a natural upper bound on how ℓ should be chosen: when $\deg \lambda + t(k-1) \geq n$ then *any* choice of λ will satisfy the t 'th key equation simply by setting $\psi_t = (\lambda R \bmod G)$, and so those equations will not further restrict λ . That gives the rough bound $\ell < n/(k-1)$.

By regarding the linearised problem as a linear system of equations, and counting available coefficients versus constraints, one arrives at an expression for the greatest number of errors we should expect to be decodable:

$$\epsilon \leq \frac{\ell}{\ell+1}n - \frac{1}{2}\ell(k-1) - \frac{\ell}{\ell+1}. \quad (3)$$

This argument does not imply that we will necessarily succeed when the bound is satisfied: the constructed system might have spurious “false solutions” that degree less than or equal to that of Λ . In such rare cases decoding might fail for fewer errors than (3). Bounding the probability that this occurs has proven difficult. We now know upper bounds when $\ell = 2, 3$ [27, 36], and Schmidt, Sidorenko, and Bossert posed a conjecture, backed by simulation, on the probability in general [36].

From (3) we can determine the value of ℓ that maximise the decoding radius. Whenever $k/n > 1/3$, one should simply choose $\ell = 1$, i.e. classical key equation decoding. Thus Power decoding is only useful for low-rate codes. Note that (3) is almost the same bound as the Sudan decoding algorithm [40], which is the Guruswami–Sudan algorithm with multiplicity 1.

Power decoding was originally described using a syndrome formulation instead of (3) [34]: we restrict ourselves to the case where 0 is not an evaluation point, and we define $S^{(t)}$ as the power series expansion of $\text{rev}(R^{(t)})/\text{rev}(G)$ truncated at $x^{n-t(k-1)-1}$, where $R^{(t)}$ is the unique polynomial of degree less than n such that $R^{(t)} \equiv R^t \pmod{G}$. Then it follows from Lemma 2.2, by the same rewriting as in Section 2.2 [27], that:

$$\text{rev}(\Lambda)S^{(t)} \equiv -\text{rev}(\Omega_t) \pmod{x^{n-t(k-1)-1}}, \quad (4)$$

where Ω_t are certain polynomials of degree at most $\epsilon - 1$ that we omit defining explicitly. It can be shown using the same rewriting that Power syndrome decoding fails if and only if Power Gao decoding fails [27].

For the Gao formulation, the linearised problem to solve is sometimes known as vector rational function reconstruction [30], and for the syndrome formulation as simultaneous Padé approximation [3] or multi-sequence shift-register synthesis [36]. Iterative algorithms with $O(\ell n^2)$ complexity can be found in [25, 36, 38]. $O^\sim(\ell^\omega n)$ algorithms are in [25, 37], while the recent [29] gives an $O^\sim(\ell^{\omega-1}n)$ algorithm.

The approach in [25] is based on computing reduced bases of carefully selected $\mathbb{F}[x]$ modules. What we describe in Section 4 to solve the new Powered key equations is a generalisation of this approach.

3. New Key Equations. In this section we describe the main result of the paper, namely a new generalisation of Power decoding where we introduce a second parameter, *the multiplicity*. The resulting relations will again be non-linear in Λ and f , and we will employ a similar linearisation strategy. We will see in Section 4 how the linear problem can be solved efficiently using a lattice basis reduction approach.

The generalised key equations are described in the following theorem:

Theorem 3.1. *For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then*

$$\begin{aligned} \Lambda^s f^t &= \sum_{i=0}^t (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{t-i} G^i \right) && \text{for } t = 1, \dots, s-1, \\ \Lambda^s f^t &\equiv \sum_{i=0}^{s-1} (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{t-i} G^i \right) \pmod{G^s} && \text{for } t = s, \dots, \ell. \end{aligned}$$

Proof. We simply rewrite

$$\begin{aligned} \Lambda^s f^t &= \Lambda^s (R + (f - R))^t \\ &= \sum_{i=0}^t \binom{t}{i} \Lambda^s (f - R)^i R^{t-i}. \end{aligned}$$

If $t < s$ then $\Lambda^s (f - R)^i = \Lambda^{s-i} \Omega^i G^i$ for each i by Lemma 2.1. This finishes the first part of the theorem.

If $t \geq s$ then for $i = s, \dots, \ell$, the summand equals $\binom{t}{i} \Lambda^{i-s} \Omega^s G^s R^{t-i}$ due to Lemma 2.1, which is 0 modulo G^s . Replacing $\Lambda^s (f - R)^i$ by $\Lambda^{s-i} \Omega^i G^i$ for $i < s$ as before gives the sought. \square

The above theorem describes ℓ equations in the unknowns $\Lambda^s, \Lambda^{s-1} \Omega, \dots, \Lambda \Omega^{s-1}$ as well as $\Lambda^s f, \dots, \Lambda^s f^\ell$. These are “key equations” in the following sense: the inner product of the first set of unknowns with a vector of known polynomials (the $\binom{t}{i} R^{t-i} G^i$) have surprisingly low degree – either immediately or reduced modulo G^s – since it is the degree of $\Lambda^s f^t$.

As with the previous key equation decoding algorithms described in Section 2, we perform the following linearisation to make the problem of finding Λ and f tractable:

Problem. *Find a vector $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) \in \mathbb{F}[x]^{s+\ell}$ with λ_0 monic and such that the following requirements are satisfied:*

$$\begin{aligned} 1a) \quad \psi_t &= \sum_{i=0}^t \lambda_i \cdot \left(\binom{t}{i} R^{t-i} G^i \right), && \text{for } t = 1, \dots, s-1 \\ 1b) \quad \psi_t &\equiv \sum_{i=0}^{s-1} \lambda_i \cdot \left(\binom{t}{i} R^{t-i} G^i \right) \pmod{G^s}, && \text{for } t = s, \dots, \ell \\ 2) \quad \deg \lambda_0 &\geq \deg \lambda_i + i, && \text{for } i = 1, \dots, s-1 \\ 3) \quad \deg \lambda_0 &\geq \deg \psi_t - t(k-1), && \text{for } t = 1, \dots, \ell. \end{aligned}$$

Clearly $\mathbf{\Lambda} = (\Lambda^s, \Lambda^{s-1} \Omega, \dots, \Lambda \Omega^{s-1}, \Lambda^s f, \dots, \Lambda^s f^\ell)$ satisfies these requirements, but there are unfortunately infinitely many other vectors satisfying them. We will therefore seek the one of least degree, i.e. where $\deg \lambda_0$ is minimal; the hope is then that this vector is $\mathbf{\Lambda}$. In that case, decoding will be completed simply by computing $f = \psi_1 / \lambda_0$.

Note that as in the simpler Power decoding of Section 2.3, the above linearisation implies a rough bound for the choices of ℓ , namely that $\ell < sn/(k-1)$. For if $t \geq sn/(k-1)$, then item 3 imposes no restrictions as long as $\deg \psi_t < sn$. But we can always choose a ϕ_t with $\deg \psi_t < sn$ and which satisfies the only other requirement on ϕ_t , namely 1b. Thus setting $\ell \geq sn/(k-1)$ would impose no further restrictions on the λ_i than would $\ell = \lceil sn/(k-1) \rceil - 1$.

Remark 1. The shape of the above equations bears a tantalising resemblance to certain approaches for solving the Interpolation phase in the Guruswami–Sudan algorithm: the $\mathbb{F}[x]$ -lattice characterisation as in [4, 20], and the (intermediate) Interpolation key equations as in [46, Eqn. (31)]. However, the Guruswami–Sudan algorithm has, a priori, nothing to do with the error locator, and the algebraic connection between the two sets of key equations is unclear. For instance, it is not known if one can easily obtain the error locator from an interpolation polynomial or vice versa.

Remark 2. The original Power decoding can be described by analogy with decoding of certain Interleaved RS codes [36]. It would be interesting to find a similar analogue for the key equations of Theorem 3.1.

4. Solving the Key Equations. We will now show how one can use $\mathbb{F}[x]$ -lattice basis reduction to find a minimal solution to Problem 3. This approach is very closely related to that of [25] for solving the powered key equations of Section 2.3. This, in turn, lends much from the Gröbner basis description for classical key equation solving by Fitzpatrick [9].

To solve Problem 3, consider first \mathcal{M} as the space of vectors $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) \in \mathbb{F}[x]^{s+\ell}$ just satisfying requirements 1a and 1b. Clearly $\Lambda \in \mathcal{M}$. It turns out that \mathcal{M} is a free $\mathbb{F}[x]$ module and in fact we know a basis for it:

Proposition 1. $\mathcal{M} = \text{Row}_{\mathbb{F}[x]}(M)$, i.e. \mathcal{M} is the $\mathbb{F}[x]$ -row space of $M \in \mathbb{F}[x]^{(\ell+1) \times (s+\ell)}$, where

$$M = \left[\begin{array}{c|cc} \mathbf{I}_{s \times s} & & N \\ \hline \mathbf{0}_{(\ell-s+1) \times s} & \mathbf{0}_{(\ell-s+1) \times (s-1)} & G^s \mathbf{I}_{(\ell-s+1) \times (\ell-s+1)} \end{array} \right],$$

where $N \in \mathbb{F}[x]^{s \times \ell}$ is the matrix whose (i, t) th entry is

$$N[i, t] = \binom{t}{i-1} R^{t-(i-1)} G^{i-1} \bmod G^s, \quad i = 1, \dots, s \text{ and } t = 1, \dots, \ell,$$

that is,

$$N = \begin{bmatrix} R & R^2 & \dots & R^{s-1} & \dots & R^\ell \\ G & 2RG & \dots & (s-1)R^{s-2}G & \dots & \ell R^{\ell-1}G \\ 0 & G^2 & \dots & \binom{s-1}{2} R^{s-3}G^2 & \dots & \binom{\ell}{2} R^{\ell-1}G \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & G^{s-1} & \dots & \binom{\ell}{s-1} RG^{s-1} \end{bmatrix} \bmod G.$$

Proof. Let \mathbf{m}_j denote the j th row of M . To show $\mathcal{M} \supset \text{Row}_{\mathbb{F}[x]}(M)$, simply note that each \mathbf{m}_j is in \mathcal{M} : for $j \leq s$ then \mathbf{m}_j corresponds in the equations of 1a and 1b to setting $\lambda_i = 0$ for all $i \neq j-1$, $\lambda_{j-1} = 1$, as well as $\psi_t = \binom{t}{j-1} R^{t-(j-1)} G^{j-1}$ for $t = 0, \dots, \ell$. This clearly leaves them satisfied. For $j > s$, \mathbf{m}_j corresponds to setting $\lambda_i = \psi_t = 0$ for all i and for $t \neq j-1$, and $\psi_{j-1} = G^s \equiv 0 \bmod G^s$.

Now for $\mathcal{M} \subset \text{Row}_{\mathbb{F}[x]}(M)$, leading to equality. For any $\mathbf{v} = (\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) \in \mathcal{M}$ then the vector $\mathbf{v}' = \sum_{i=1}^s \lambda_{i-1} \mathbf{m}_i$ agrees with \mathbf{v} in the first $2s - 1$ positions. For the remaining positions $h = 2s, \dots, s + \ell$ of \mathbf{v}' , the position is congruent to ψ_{h-s} modulo G^s . Therefore there exists $q_s, \dots, q_\ell \in \mathbb{F}[x]$ such that $\mathbf{v} = \mathbf{v}' + \sum_{i=s+1}^{\ell+1} q_i \mathbf{m}_i$, and thus $\mathcal{M} \subset \text{Row}_{\mathbb{F}[x]}(M)$. \square

To find a minimal solution to Problem 3, we should therefore seek a vector $\mathbf{v} = (\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) \in \text{Row}_{\mathbb{F}[x]}(M)$ such that:

1. $\deg \lambda_0 \geq \deg \lambda_i + i$
2. $\deg \lambda_0 + t(k-1) \geq \deg \psi_t$
3. $\deg \lambda_0$ is minimal under these constraints and λ_0 is monic.

These goals turn out to be achievable by finding another matrix whose rows span \mathcal{M} but which is in *weak Popov form*. This form was introduced by Mulders and Storjohann in [23] as a slightly stronger form than *row reduced* [16, p. 380], but which exactly allows to argue about restrictions such as the degree inequalities above. The rows of a matrix in weak Popov form are also a Gröbner basis for the module \mathcal{M} for the term-over-position ordering; however we will stay with the matrix language in this exposition. Our strategy is very similar to finding short vectors in modules by computing a row reduced basis, see e.g. [43, Problem 16.12]. In this settings, *shifts* as we will use have also been considered, see e.g. [48].

Definition 4.1. The *leading position* of a non-zero vector $\mathbf{v} \in \mathbb{F}[x]^m$, written $\text{LP}(\mathbf{v})$, is the right-most entry in \mathbf{v} with maximal degree among the entries of \mathbf{v} . A matrix $V \in \mathbb{F}[x]^{m_1 \times m_2}$ is in *weak Popov form* if the leading positions of the non-zero rows are all different.

The following is an easy generalisation of [23, Lemma 8.1], and also appears in [25]; we give the short proof here for completeness.

Proposition 2. Let $V \in \mathbb{F}[x]^{m_1 \times m_2}$ be a basis in weak Popov form of a module \mathcal{V} . Any non-zero $\mathbf{b} \in \mathcal{V}$ satisfies $\deg \mathbf{v} \leq \deg \mathbf{b}$ where \mathbf{v} is the row of V with $\text{LP}(\mathbf{v}) = \text{LP}(\mathbf{b})$. If a leading position is not represented by a row in V , then no vector in \mathcal{V} has that leading position.

Proof. Let $\mathbf{u} \in \mathcal{V}$ be non-zero, and so there exists $a_1, \dots, a_\ell \in \mathbb{F}[x]$ not all zero such that $\mathbf{u} = \sum_{i=1}^{m_1} a_i \mathbf{v}_i$ where the \mathbf{v}_i are the rows of V . The \mathbf{v}_i all have different leading position, so the $a_i \mathbf{v}_i$ also have different leading position among those i where $a_i \neq 0$. Note that for any two $\mathbf{u}_1, \mathbf{u}_2$ with $\text{LP}(\mathbf{u}_1) \neq \text{LP}(\mathbf{u}_2)$, then $\mathbf{u}_1 + \mathbf{u}_2$ has the same degree and leading position of either \mathbf{u}_1 or \mathbf{u}_2 . Applied inductively, that implies that there is an i such that $\text{LP}(\mathbf{u}) = \text{LP}(a_i \mathbf{v}_i)$ and $\deg \mathbf{u} = \deg(a_i \mathbf{v}_i) \geq \deg \mathbf{v}_i$. \square

The above proposition means that if V is a basis in weak Popov form of some module \mathcal{V} , then the rows of V have minimal degree for each possible leading position. So we can use the weak Popov form to find small-degree vectors which has the greatest degree polynomial on a specific index. Our degree restrictions single out λ_0 as somehow “leading”, but under integral *shifts*, e.g. of the form $\deg \lambda_0 + t(k-1) \geq \deg \psi_t$. We will handle these shifts by incorporating them directly into the module.

First, we introduce non-negative variables $\mu_0, \dots, \mu_{s-1}, \eta_1, \dots, \eta_\ell \in \mathbb{N}_0$ as

$$\mu_0 = 1 + \ell(k-1) \quad \mu_i = i + \ell(k-1), \quad i > 0 \quad \eta_t = (\ell - t)(k-1), \quad \forall t. \quad (5)$$

Our degree restrictions now read

$$\deg \lambda_0 + \mu_0 > \deg \lambda_i + \mu_i, \quad i = 1, \dots, s-1, \quad (6)$$

$$\deg \lambda_0 + \mu_0 > \deg \psi_t + \eta_t, \quad t = 1, \dots, \ell. \quad (7)$$

Notice that a vector $\mathbf{v} = (\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell)$ satisfies these degree restrictions if and only if $\text{LP}(\mathbf{v}D) = 1$, where

$$D = \text{diag}(x^{\mu_0}, \dots, x^{\mu_{s-1}}, x^{\eta_1}, \dots, x^{\eta_\ell}). \quad (8)$$

Consider therefore the module $\hat{\mathcal{M}} = \{\mathbf{v}D \mid \mathbf{v} \in \mathcal{M}\}$, spanned by the rows of $\hat{M} = MD$. We arrive at:

Corollary 1. *Let $\hat{B} = BD$ be a basis of $\hat{\mathcal{M}}$ and in weak Popov form, and let $\hat{\mathbf{v}}$ be the row of \hat{B} with leading position 1. Then $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) = \gamma \hat{\mathbf{v}} D^{-1}$ constitutes a solution to Problem 3 such that $\deg \lambda_0$ is minimal, where $\gamma \in \mathbb{F}^*$ is chosen such that λ_0 is monic.*

Proof. Note that $\hat{\mathbf{v}}$ is well defined since \hat{B} is a square, full-rank matrix in weak Popov form and therefore must have a row of leading position 1. By Proposition 2, $\hat{\mathbf{v}}$ must have minimal degree among vectors in $\hat{\mathcal{M}}$ with leading position 1. The above discussion then implies that $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell)$ satisfies the degree constraints of Problem 3, and that λ_0 has minimal degree among the first term of vectors satisfying these constraints. This vector also satisfies the congruence constraint of Problem 3 since it is in \mathcal{M} . \square

Note that any basis of $\hat{\mathcal{M}}$ seen as a matrix must be $\mathbb{F}[x]$ -divisible on the right by D . So to find a \hat{B} satisfying Corollary 1, we need only compute $\hat{B} \sim \hat{M}$ such that \hat{B} is in weak Popov form. By \sim we mean left unimodular equivalence, i.e. $A \sim B$ for two $A, B \in \mathbb{F}[x]^{m_1 \times m_2}$ if there exists an invertible matrix $U \in \mathbb{F}[x]^{m_1 \times m_2}$ such that $A = UB$. The $\mathbb{F}[x]$ row spaces of matrices A and B are the same if and only if $A \sim B$.

The complete decoding algorithm, with the weak Popov form computation as a black box, is given as Algorithm 1.

Proposition 3. *Algorithm 1 is correct.*

Proof. For any codeword $\hat{\mathbf{c}} \in \mathcal{C}$, there is an associated error $\hat{\mathbf{e}} = \mathbf{r} - \hat{\mathbf{c}}$ and thus error locator $\hat{\Lambda}$ and error evaluator $\hat{\Omega}$. These satisfy Theorem 3.1 and therefore induce a solution to Problem 3. The first component of this solution is $\hat{\Lambda}^s$.

By Corollary 1, the $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell)$ computed in Line 4 is a solution to Problem 3 where the first component λ_0 has minimal degree. No codeword can therefore have distance less than $\deg \lambda_0 / s$ from \mathbf{r} , since it would induce a solution to Problem 3 with smaller degree than $\deg \lambda_0$ on the first component.

If fail is not returned in Line 5 then the computed f satisfies $\deg f < k$, since $\deg \psi_1 \leq \deg \lambda_0 + (k-1)$. Thus $\text{ev}(f) \in \mathcal{C}$. Since $\text{ev}(f)$ is only returned if its distance to \mathbf{r} is exactly $\deg \lambda_0 / s$, this must be a codeword of minimal distance to \mathbf{r} . \square

Algorithm 1 leaves unspecified how to compute \hat{B} , i.e. how to compute a basis of $\hat{\mathcal{M}}$ in weak Popov form. Since we are initially given a different basis of $\hat{\mathcal{M}}$, namely \hat{M} , the problem is that of finding a matrix which is unimodular equivalent to \hat{M} but in weak Popov form. This problem is well-studied in computer algebra, and several algorithms exist which solve this problem directly [2, 23] or through a related

Algorithm 1 Efficient Power Decoding with Multiplicities**Input:** $\mathbf{r} \in \mathbb{F}^n$, $s, \ell \in \mathbb{Z}_+$.**Output:** $\tilde{\mathbf{c}} \in \mathcal{C}$ such that $\text{dist}(\tilde{\mathbf{c}}, \mathbf{r})$ is minimal among codewords in \mathcal{C} , or fail

- 1 Compute R such that $R(\alpha_i) = r_i$.
- 2 Construct M as in Proposition 2 and $\hat{M} \triangleq MD$, where D is as in (8).
- 3 Compute \hat{B} in weak Popov form such that $\hat{B} \sim \hat{M}$.
- 4 Let \hat{v} be the row of \hat{B} with $\text{LP}(\hat{v}) = 1$. Let $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell) \triangleq \gamma \hat{v} D^{-1}$, with $\gamma \in \mathbb{F}^\star$ such that λ_0 is monic.
- 5 If λ_0 divides ψ_1 , let $f \triangleq \psi_1 / \lambda_0$. Otherwise fail.
- 6 If $\text{dist}(\mathbf{r}, \text{ev}(f)) = \deg \lambda_0 / s$ then return $\text{ev}(f)$. Otherwise fail.

form [11, 13, 33, 48]. Recall that in complexity bounds, ω denotes the exponent of matrix multiplication and $M(n)$ the cost of multiplying together two polynomials of degree at most n . Then we have:

Proposition 4. *Given a matrix $A \in \mathbb{F}[x]^{m_1 \times m_2}$ there exists an algorithm to compute a matrix $B \in \mathbb{F}[x]^{m_1 \times m_2}$ in weak Popov form and $B \sim A$ in complexity $O^\sim(m^\omega M(\deg A))$ [13], as well as one for computing it in complexity $O(m^3 \deg A^2)$ [23], where $m = \max(m_1, m_2)$.*

Corollary 2. *Algorithm 1 can be implemented with asymptotic complexity either $O^\sim(\ell^\omega sn)$ or $O(\ell^3 s^2 n^2)$.*

Proof. G^i can be precomputed for $i = 1, \dots, s$. Computing R can be done in $O(M(n))$ using Lagrange interpolation, see e.g. [43, p. 297]. We compute all $R^i \bmod G^s$ for $i = 1, \dots, \ell$ in time $O(\ell s M(n))$. Constructing M then requires $O(\ell^2)$ elements that are scalings of the product of two known polynomials of degree at most sn , requiring $O(\ell^2 M(sn))$. Clearly, the remaining lines apart from Line 3 are cheaper. We have $\deg \hat{M} \in O(sn)$ since $\ell(k-1) < sn$. Thus the complexities for executing line Line 3 given by Proposition 4 dominates, whether or not a fast polynomial multiplication algorithm is used (i.e. whether setting $M(N) = O^\sim(N)$ or $M(N) = N^2$). \square

4.1. A Punctured Module by Ignoring Error-Evaluators. It is possible to obtain a smaller matrix than M which will provide us with equally good decoding performance. This results in a faster decoding algorithm, though the asymptotic complexity remains unchanged. A second benefit of the optimisation is that it makes it easier for us to reason on the decoding radius of the algorithm in Section 5.

The optimisation is based on two observations: Firstly, for decoding we do not need to know $\lambda_1, \dots, \lambda_s$ since the later steps of the algorithm only uses λ_0 and ψ_0 . Secondly, in the failure probability bound that we derive in Section 6 (for $s = 2$ and $\ell = 3$) the degree restrictions $\deg \lambda_0 > \deg \lambda_i + i$ are not used, and thus perhaps they have little influence on the failure probability in general.

In the lattice view, this means that we can simply delete columns 2 to s in both M and in D . Bringing this smaller matrix $\tilde{M}\tilde{D} \in \mathbb{F}[x]^{(\ell+1) \times (\ell+1)}$ to weak Popov form will still solve for the remaining conditions. As a result, we obtain a vector $(\lambda_0, \psi_1, \dots, \psi_\ell)$ hopefully equal to $(\Lambda^s, \Lambda^s f, \dots, \Lambda^s f^\ell)$.

Note that the punctured matrix \check{M} takes the form:

$$\check{M} = \left[\begin{array}{c|cccc|cccc} 1 & R & R^2 & \dots & R^{s-1} & R^s & \dots & R^\ell \\ 0 & G & 2RG & \dots & (s-1)R^{s-2}G & sR^{s-1}G & \dots & \ell R^{\ell-1}G \\ 0 & 0 & G^2 & \dots & \binom{s-1}{2}R^{s-3}G^2 & \binom{s}{2}R^{s-2}G^2 & \dots & \binom{\ell}{2}R^{\ell-1}G \\ & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & G^{s-1} & sRG^{s-1} & \dots & \binom{\ell}{s-1}RG^{s-1} \\ \hline & \mathbf{0}_{(\ell-s+1) \times s} & & & & G^s \mathbf{I}_{(\ell-s+1) \times (\ell-s+1)} & & \end{array} \right].$$

That is, \check{M} is a square, upper-diagonal matrix of full rank.

It is theoretically not clear whether working with \check{M} instead of M could result in an increased failure probability. However simulations indicate this is not the case. Also, as already mentioned, the failure probability bound derived in Section 6 applies for when working with M .

5. Decoding Radius. We are now in a position to discuss how many errors the method will usually cope with. When calling this a “decoding radius” we need to be wary: indeed, the method *will* fail for certain received words whenever the number of errors is at least $d/2$, and this is unavoidable since it is a unique decoding algorithm. Therefore, “decoding radius” really involves two parts: 1) how many errors should we at most expect to be able to correct; and 2) what is the probability that we will fail within this number of errors.

Perhaps not surprisingly, the latter question is much more difficult than the former. In this section we will derive an upper bound on error correction. In the next section we discuss the failure behaviour when fewer errors than this has occurred.

The decoding radius upper bound that we will derive is based on bringing $\check{M}\check{D}$ from Section 4.1 to weak Popov form. Recall that what we are essentially doing is finding the lowest degree vector in $\text{Row}_{\mathbb{F}[x]}(\check{M}\check{D})$ which has first position as leading. The outcome of imposing the leading position requirement is sensitively dependent on the module \mathcal{M} , but bounding the size of the lowest degree vector *overall* is easy:

Proposition 5. *A vector $\mathbf{v} = (\check{\lambda}_0, \check{\psi}_1, \dots, \check{\psi}_\ell)$ such that $\mathbf{v}\check{D}$ is a minimal degree vector in $\text{Row}_{\mathbb{F}[x]}(\check{M}\check{D})$ satisfies*

$$\frac{\deg \check{\lambda}_0}{s} \leq \tau_{\text{Pow}}(s, \ell) \triangleq \frac{2\ell - s + 1}{2(\ell + 1)}n - \frac{\ell}{2s}(k - 1) - \frac{\ell}{s(\ell + 1)}. \quad (9)$$

If $\epsilon > \tau_{\text{Pow}}(s, \ell)$ then $\deg \Lambda > \deg \check{\lambda}_0/s$.

Proof. Let \check{B} be a matrix such that $\check{B}\check{D}$ is in weak Popov form and unimodular equivalent to $\check{M}\check{D}$. Recall that $\check{D} = \text{diag}(x^{\mu_0}, x^{\eta_1}, \dots, x^{\eta_\ell})$.

We know from Proposition 2 that since \check{M} is square then for any leading position, $\check{B}\check{D}$ contains a vector of minimal degree in $\text{Row}_{\mathbb{F}[x]}(\check{M}\check{D})$ for that leading position, and therefore it also contains a row with minimal degree overall, i.e. with the same degree as $\mathbf{v}\check{D}$. Now, it is easy to see that since $\check{B}\check{D}$ is in weak Popov form, then $\deg \det(\check{B}\check{D}) = \sum_{i=1}^{\ell+1} \deg \mathbf{b}_i$, where \mathbf{b}_i are the rows of $\check{B}\check{D}$ (see e.g. [16, p. 384] since weak Popov form implies row reduced). Thus, not all the \mathbf{b}_i can have degree greater than $\frac{1}{\ell+1} \deg \det(\check{B}\check{D})$, and so this bounds the degree of $\mathbf{v}\check{D}$.

Clearly $\det \check{B} = \det \check{M}$. Since \check{M} is an upper-diagonal matrix, we can therefore easily compute its determinant:

$$\deg \det \check{M} = \deg \left(G^{s(\ell-s+1)} \prod_{i=1}^{s-1} G^i \right) = (s\ell - \binom{s}{2})n .$$

Also $\deg \det \check{D} = \mu_0 + \sum_t \eta_t = \binom{\ell+1}{2}(k-1) + 1$. Thus we have

$$\begin{aligned} \deg \check{\lambda}_0 + \mu_0 &\leq \deg(\mathbf{v}\check{D}) \\ &\leq \frac{1}{\ell+1}((s\ell - \binom{s}{2})n + \binom{\ell+1}{2}(k-1) + 1) , \end{aligned}$$

which rewrites into the sought bound. \square

When solving the key equations, we will seek a minimal degree vector in $\text{Row}_{\mathbb{F}[x]}(\check{M}\check{D})$ which has leading position 1. We are then hoping that the first element of this vector equals Λ^s . Since the minimal degree vector overall in the row space might not have leading position 1, the above corollary doesn't quite state that we will surely fail in decoding when $\epsilon > \tau_{\text{Pow}}(s, \ell)$. However, it is natural to suspect that most likely, the minimal degree vector with leading position 1 has degree quite close to the minimal degree overall. Therefore, we might *expect* to fail. This intuition is also backed by simulation, see Section 7.

We can relate $\tau_{\text{Pow}}(s, \ell)$ to something very well known:

Corollary 3. *Denote the maximal decoding radius of the Guruswami–Sudan algorithm on \mathcal{C} with multiplicity s and list size ℓ by $\tau_{\text{GS}}(s, \ell)$. Then*

$$\tau_{\text{GS}}(s, \ell) = \frac{2\ell - s + 1}{2(\ell + 1)}n - \frac{\ell}{2s}(k-1) = \tau_{\text{Pow}}(s, \ell) + \frac{\ell}{s(\ell + 1)} .$$

(see e.g. [31, Lemma 9.5]).

Taken over all s and ℓ , the decoding radius of Guruswami–Sudan describes a curve $J(n, d) = n - \sqrt{n(n-d)}$, often called the Johnson radius after [15]. For any integer $\tau < J(n, d)$ there exists infinitely many choices of s, ℓ such that $\tau = \lfloor \tau_{\text{GS}}(s, \ell) \rfloor$. Thus, by Corollary 3, Power decoding is similarly bounded by the Johnson radius. The corollary even tells us more: if we choose exactly the same s and ℓ as in the Guruswami–Sudan, then Power decoding will decode up to the same radius or at most 1 less.

For the Guruswami–Sudan, good, closed-form expressions for small s and ℓ given the code and τ can be found in [26, p. 53]. These therefore immediately apply to Power decoding as well.

6. Failure Behaviour. We will move on to investigate how Power decoding fails when at most $\tau_{\text{Pow}}(s, \ell)$ errors occur. There are two ways in which Algorithm 1 can give an unwanted answer: firstly, the algorithm can return *fail*; or secondly, the algorithm can return a different codeword than the sent one. For a specific sent codeword \mathbf{c} and received word \mathbf{r} , we say that Power decoding *fails* if one of the two following conditions are satisfied:

1. Algorithm 1 returns *fail*.
2. There exists $\mathbf{c}' \in \mathcal{C}$, $\mathbf{c}' \neq \mathbf{c}$, and such that $\text{dist}(\mathbf{r}, \mathbf{c}') \leq \text{dist}(\mathbf{r}, \mathbf{c})$.

Recall that when Algorithm 1 does not return *fail*, it always returns a codeword of minimal distance to the received. So if neither of the above conditions are satisfied, Algorithm 1 returns the correct answer. Contrarily, if only item 2 above is satisfied and $\text{dist}(\mathbf{r}, \mathbf{c}') = \text{dist}(\mathbf{r}, \mathbf{c})$, and \mathbf{c} might still be correctly returned. This, however,

depends rather arbitrarily on exactly which matrix \hat{B} is computed by the weak Popov form algorithm. For the sake of a cleaner definition, we therefore consider this possibility as a failure as well.

We will begin with showing that the error vector alone determines whether the method succeeds. This drastically simplifies further examinations on the failure behaviour. It allows us first to show the—quite expected—property that the method never fails when fewer than $d/2$ errors occur. Secondly, it allows us to give a closed upper bound on the failure probability when $(s, \ell) = (2, 3)$. Lastly, we discuss the relation between Power decoding failing and having multiple close codewords to the received word.

Proposition 6. *Power decoding succeeds for some received word \mathbf{r} if and only if it succeeds for $\mathbf{r} + \hat{\mathbf{c}}$ where $\hat{\mathbf{c}}$ is any codeword.*

Proof. If Power decoding fails for \mathbf{r} as received word, this is because there exist $\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell \in \mathbb{F}[x]$ which solve Problem 3, and where $\lambda_0 \neq \Lambda^s$ and $\deg \lambda_0 \leq \deg \Lambda^s$. Assume this is the case. Let \hat{R} be the Lagrange interpolant corresponding to $\mathbf{r} + \hat{\mathbf{c}}$ as received word, i.e. $\hat{R} = R + \hat{f}$ where $\hat{f} = \text{ev}^{-1}(\hat{\mathbf{c}})$ and $\deg \hat{f} < k$. We will show that there exist $\hat{\psi}_1, \dots, \hat{\psi}_\ell \in \mathbb{F}[x]$ such that the $\lambda_i, \hat{\psi}_t$ form a solution to Problem 3 for \hat{R} in place of R . Therefore, Power decoding will also fail for $\mathbf{r} + \hat{\mathbf{c}}$ as received word.

Consider for $t = 1, \dots, \ell$ the following expansion:

$$\begin{aligned} & \sum_{i=0}^{\min(t, s-1)} \lambda_i \cdot \binom{t}{i} \hat{R}^{t-i} G^i \\ &= \sum_{i=0}^{\min(t, s-1)} \lambda_i \binom{t}{i} \left(\sum_{h=0}^{t-i} \binom{t-i}{h} R^{t-i-h} \hat{f}^h \right) G^i \\ &= \sum_{h=0}^t \hat{f}^h \sum_{i=0}^{\min(t-h, s-1)} \lambda_i \binom{t}{i} \binom{t-i}{h} R^{t-i-h} G^i. \end{aligned}$$

Note now that $\binom{t}{i} \binom{t-i}{h} = \binom{t}{h} \binom{t-h}{i}$. Therefore, the above equals

$$\begin{aligned} & \sum_{h=0}^t \binom{t}{h} \hat{f}^h \sum_{i=0}^{\min(t-h, s-1)} \lambda_i \binom{t-h}{i} R^{t-i-h} G^i \\ & \equiv \sum_{h=0}^t \binom{t}{h} \hat{f}^h \psi_{t-h}, \end{aligned}$$

where we by “ \equiv ” mean $=$ when $t < s$ and congruent modulo G^s when $t \geq s$. We set $\hat{\psi}_t$ as the last expression above. By hypothesis, $\deg \psi_{t-h} - (t-h)(k-1) < \deg \lambda_0$. Since $\deg \hat{f} < k$ we therefore get $\deg \hat{\psi}_t - t(k-1) < \deg \lambda_0$.

This means the λ_i, ψ_t indeed form a solution to Problem 3 for \hat{R} , as we set out to prove.

The proved implication can immediately be applied in the other direction since $-\hat{\mathbf{c}}$ is a codeword, showing the bi-implication. \square

We now prove that Power decoding always succeeds in half-the-minimum distance decoding. The proof is surprisingly technical since we need to keep a handle on the key equations simultaneously.

Proposition 7. *If fewer than $d/2$ errors occur, then Power decoding succeeds.*

Proof. By Proposition 6, we can assume that $\mathbf{0}$ was sent. By Lemma 2.1 we then have $R = -\Omega\Upsilon$, where $\Upsilon = G/\Lambda$.

Assume contrary to the proposition that Power decoding has failed. That means there exists $(\lambda_0, \dots, \lambda_{s-1}, \psi_1, \dots, \psi_\ell)$ which solve Problem 3, and where $\lambda_0 \neq \Lambda^s$ and $\deg \lambda_0 \leq \deg \Lambda^s$. We will inductively establish $P(t)$ for $t = 0, \dots, s-1$, where $P(t)$ is the assertion

$$P(t) : \quad \Lambda^{t+1-i} \mid \lambda_i \text{ and } \psi_{s-i} = 0 \text{ for } i = 0, \dots, t .$$

For $t = s-1$, $P(t)$ implies $\Lambda^s \mid \lambda_0$, which contradicts the minimality of λ_0 , finishing the proof.

For the case $P(0)$, we need to prove that $\Lambda \mid \lambda_0$ and $\psi_s = 0$. Consider the s 'th key equation of Problem 3 which is satisfied by the λ_i and ψ_s :

$$\psi_s \equiv \sum_{i=0}^{s-1} \binom{s}{i} \lambda_i R^{s-i} G^i \pmod{G^s} . \quad (10)$$

Υ^s divides each term of the summand, as well as the modulus G^s , and so it must divide ψ_s . However, we have

$$\deg \psi_s \leq \deg \lambda_0 + s(k-1) \leq s\epsilon + s(k-1) < s(n-\epsilon) ,$$

where the last inequality holds since $2\epsilon < n - k + 1$. Thus $\psi_s = 0$.

Returning to (10), we can then conclude $\Lambda \mid \lambda_0 R^s$, since Λ divides every other term in the sum as well as the modulus. This implies $\Lambda \mid \lambda_0$ since $\gcd(\Lambda, R) = 1$.

For the inductive step, assuming $P(t-1)$ we will prove $P(t)$ for $1 \leq t < s$. Consider now the $(s-t)$ 'th key equation, i.e.

$$\psi_{s-t} = \sum_{i=0}^{s-t} \binom{s-t}{i} \lambda_i R^{s-t-i} G^i .$$

Similar to before, Υ^{s-t} divides every term of the sum, so it divides ψ_{s-t} . By $P(t-1)$ then $\Lambda^{t-i} \mid \lambda_i$ for $i = 0, \dots, t-1$, and therefore $\Lambda^t \mid \lambda_i R^{s-t-i} G^i$. This implies $\Lambda^t \mid \psi_{s-t}$ and hence $\Upsilon^{s-t} \Lambda^t \mid \psi_{s-t}$. But now we have

$$\deg \psi_{s-t} \leq \deg \lambda_0 + (s-t)(k-1) \leq s\epsilon + (s-t)(k-1) < (s-t)(n-\epsilon) + t\epsilon ,$$

which means $\psi_{s-t} = 0$.

It remains to show that $\Lambda^{t+1-i} \mid \lambda_i$ for $i = 0, \dots, t-1$. For $j = 1, \dots, t$, multiply the $(s-j)$ 'th key equation with R^j and relax it to a congruence modulo G^s . We obtain $t+1$ homogeneous linear equations in $\lambda_i R^{s-i} G^i$ of the form:

$$0 \equiv \sum_{i=0}^{\min(s-1, s-j)} \binom{s-j}{i} (\lambda_i R^{s-i} G^i) \pmod{G^s} , \quad j = 0, \dots, t .$$

Subtracting the j th equation from the $(j-1)$ st for $j = 1, \dots, t$, we eliminate λ_0 and get

$$0 \equiv \sum_{i=1}^{s-1} \binom{s-j}{i-1} (\lambda_i R^{s-i} G^i) \pmod{G^s} , \quad j = 1, \dots, t .$$

This can be continued to get a series of equation systems, that is, for $t' = 1, \dots, t$, we have a system:

$$0 \equiv \sum_{i=t'}^{s-1} \binom{s-j}{i-t'} (\lambda_i R^{s-i} G^i) \pmod{G^s}, \quad j = t', \dots, t.$$

For $t' = t$, the system (which is one equation) implies that $\Lambda^{t+1} \mid \lambda_t R^{s-t} G^t$ since Λ^{t+1} divides all the sum's other terms and the modulus, and this implies $\Lambda \mid \lambda_t$. We can now go to the $t' = t-1$ system and regard any of the two equations, and we conclude similarly that $\Lambda^{t+1} \mid \lambda_{t-1} R^{s-t+1} G^{t-1}$ since Λ^{t+1} now is seen to divide all other terms of the sum as well as the modulus. This implies $\Lambda^2 \mid \lambda_{t-1}$. Continuing with decreasing t' we can iteratively conclude $\Lambda^{t+1-t'} \mid \lambda_{t'}$.

This finishes the induction step, establishing $P(t)$ for $t = 0, \dots, s-1$. As mentioned, this implies a contradiction, finishing the proof. \square

We are now in a position to bound the probability that Power decoding fails if errors of a given weight are drawn uniformly at random, for the case $(s, \ell) = (2, 3)$. Note that by Proposition 5, then

$$\tau_{\text{Pow}}(2, 3) = \frac{5}{8}n - \frac{3}{4}(k-1) - \frac{3}{8},$$

so these parameters allow the decoder to improve upon both half-the-minimum distance and the original Power decoding whenever the rate is between $1/6$ and $1/2$, for long enough codes.

Proposition 8. *Let $q = |\mathbb{F}|$. Whenever $d/2 \leq \epsilon < \tau_{\text{Pow}}(2, 3)$, the probability that Power decoding fails is upper bounded by*

$$\begin{cases} 4(q^{-8})^{(\tau_{\text{Pow}}(2,3)-\epsilon)-(0.29\epsilon/\log q-1/4)} & \text{when } \epsilon \geq \frac{3}{5}n - \frac{4}{5}(k-1) \\ 4(q^{-1})^{(d+3(k-1))/5-(\epsilon \log 10 / \log q + 1)} & \text{when } \epsilon < \frac{3}{5}n - \frac{4}{5}(k-1) \end{cases}.$$

Proof. By Proposition 6, we can consider the probability over the choice of error vector, and simply bound the failure probability when the sent codeword was $\mathbf{0}$. Since we know by Proposition 7 that the failure probability is zero when $\epsilon < d/2$, then we can also assume $\epsilon \geq d/2$.

Fix now the number of errors ϵ and error positions \mathcal{E} , implying a specific Λ . For a given error $\mathbf{e} = \mathbf{r}$ with these non-zero positions, we will call \mathbf{r} , or R , “bad” if for R there exist λ_i, ψ_t solving Problem 3 and such that $\lambda_0 \neq \Lambda^s$ while $\deg \lambda_0 \leq \deg \Lambda^s$. Consequently, Power decoding fails only for bad error-values. Denote by $S_\Lambda \subset \mathbb{F}[x]$ the set of bad R . We will give an upper bound N on the size of S_Λ and so $N/(q-1)^\epsilon$ bounds the probability that for the fixed error positions, Power decoding fails (since for each position, we have $q-1$ choices of an error value). N will turn out to be independent of the choice of Λ , and thus $N/(q-1)^\epsilon$ is a bound on the probability that Power decoding fails for any error of weight ϵ .

By assumption, the following equations are satisfied:

$$\begin{aligned} \psi_1 &= \lambda_0 R + \lambda_1 G, \\ \psi_2 &\equiv \lambda_0 R^2 + 2\lambda_1 R G \pmod{G^2}, \\ \psi_3 &\equiv \lambda_0 R^3 + 3\lambda_1 R^2 G \pmod{G^2}. \end{aligned}$$

Since $R(\alpha_i) = 0$ whenever $i \notin \mathcal{E}$, then $\Upsilon \mid R$ where $\Upsilon = G/\Lambda$. Thus the above implies $\Upsilon \mid \psi_1$ and $\Upsilon^2 \mid \psi_t$ for $t = 2, 3$. Furthermore, we can conclude that

$g \triangleq \gcd(\psi_t, \Lambda)$ is the same for all t , since $g = \gcd(\lambda_0, \Lambda)$. The regular form of the above three equations allows eliminating λ_0 and obtain:

$$\begin{aligned}\psi_2 - R\psi_1 &\equiv \lambda_1 RG \pmod{G^2}, \\ \psi_3 - R\psi_2 &\equiv \lambda_1 R^2 G \pmod{G^2}.\end{aligned}$$

From this we first note that $G \mid (\psi_2 - R\psi_1)$. We will use this fact momentarily. With the two above equations we continue to eliminate λ_1 and rewrite:

$$\begin{aligned}\psi_3 - R\psi_2 - R(\psi_2 - R\psi_1) &\equiv 0 \pmod{G^2} && \Longleftrightarrow \\ R^2\psi_1 - 2R\psi_2 + \psi_3 &\equiv 0 \pmod{G^2} && \implies \\ R^2\psi_1^2 - 2R\psi_1\psi_2 + \psi_1\psi_3 &\equiv 0 \pmod{G^2} && \Longleftrightarrow \\ (R\psi_1 - \psi_2)^2 + \psi_1\psi_3 &\equiv \psi_2^2 \pmod{G^2}.\end{aligned}$$

But we concluded just before that $G \mid (R\psi_1 - \psi_2)$ so $(R\psi_1 - \psi_2)^2 \equiv 0 \pmod{G^2}$. This leaves the simple relation

$$\begin{aligned}\psi_2^2 &\equiv \psi_1\psi_3 \pmod{G^2} && \Longleftrightarrow \\ \check{\psi}_2^2 \Upsilon &\equiv \check{\psi}_1\check{\psi}_3 \pmod{\Lambda^2},\end{aligned}\tag{11}$$

where $\check{\psi}_t \triangleq \psi_t / \Upsilon^{\min(2,t)}$, and is a polynomial by our earlier observations. Thus, whenever R is bad, there is a triple $(\check{\psi}_1, \check{\psi}_2, \check{\psi}_3) \in \mathbb{F}[x]$ satisfying the above relation as well as

$$\deg \check{\psi}_t \leq d_t \triangleq 2\epsilon + t(k-1) - \min(2, t)(n-\epsilon).\tag{12}$$

We will count the number of such triples momentarily. However, to thusly bound the number of bad error values, we have to determine how many different R could have the same triple. Recall that determining R up to congruence modulo Λ suffices, since this determines the error values. However, by our previous observation we have

$$\begin{aligned}R\psi_1 &\equiv \psi_2 \pmod{G} && \Longleftrightarrow \\ R\check{\psi}_1 &\equiv \check{\psi}_2 \Upsilon \pmod{\Lambda} && \Longleftrightarrow \\ R &\equiv (\check{\psi}_2 \Upsilon / g)(\check{\psi}_1 / g)^{-1} \pmod{\Lambda/g}.\end{aligned}$$

This means that for a given triple $(\check{\psi}_t)_t$, having $\gcd(\check{\psi}_t, \Lambda) = g$, there can be at most $q^{\deg g}$ possible choices of R .

To bound the number of bad error values N for this given Λ , we will therefore perform a weighted count of all triples satisfying (11) and (12), where a triple is counted with weight $q^{\deg g}$, where g is a divisor of Λ dividing all the $\check{\psi}_t$:

$$\begin{aligned}N &\leq \sum_{g|\Lambda} q^{\deg g} \left| \left\{ (\check{\psi}_t)_t \in \mathbb{F}[x]^3 \mid g \mid \check{\psi}_t, \deg \check{\psi}_t \leq d_t, \Upsilon \check{\psi}_2^2 \equiv \check{\psi}_1 \check{\psi}_3 \pmod{\Lambda^2} \right\} \right| \\ &= \sum_{g|\Lambda} q^{\deg g} \left| \left\{ (\check{\psi}_t)_t \in \mathbb{F}[x]^3 \mid \deg \check{\psi}_t \leq d_t - \deg g, \Upsilon \check{\psi}_2^2 \equiv \check{\psi}_1 \check{\psi}_3 \pmod{(\Lambda/g)^2} \right\} \right|.\end{aligned}$$

Let T_g be the set inside the last sum. We use Lemma 6.1 (see below) to upper bound $|T_g|$, for any choice of g : setting $A = (\Lambda/g)^2$, $B = \Upsilon$ and $K_t = d_t - \deg g$ in that lemma, we get

$$|T_g| \leq 2^{\gamma+2\epsilon-2\deg g} q^{4\epsilon-(2n-2(k-1))+1+\max(0,\gamma)-\deg g},$$

where $\gamma = 5\epsilon - (3n - 4(k - 1))$. This is only dependent on the *degree* of g . For each choice of $\deg g$, we can select g in $\binom{\epsilon}{\deg g}$ ways since $g \mid \Lambda$. For the case $\gamma \geq 0$, this gives us:

$$N \leq q^{\epsilon + 8(\epsilon - \tau_{\text{Pow}}(2,3)) - 2} 2^\gamma \sum_{t=0}^{\epsilon} \binom{\epsilon}{t} 4^{\epsilon-t},$$

since $8\epsilon - (5n - 6(k - 1) - 3) = 8(\epsilon - \tau_{\text{Pow}}(2,3))$.

This can now be simplified. Firstly $\sum_{t=0}^{\epsilon} \binom{\epsilon}{t} 4^{\epsilon-t} = (4+1)^\epsilon$. Secondly, $\gamma \leq \epsilon$, as can be seen as follows: since $\epsilon < \tau_{\text{Pow}}(2,3)$ then $4\epsilon < \frac{5}{2}n - 3(k - 1)$. Inserting in the expression for γ , we get that $\gamma \leq \epsilon - (n/2 - (k - 1))$. But since we assumed $d/2 \leq \tau_{\text{Pow}}(2,3)$ then $k - 1 \leq n/2$ which means $\gamma \leq \epsilon$. Therefore:

$$N \leq q^{\epsilon + 8(\epsilon - \tau_{\text{Pow}}(2,3)) - 2} 10^\epsilon \quad (\gamma \geq 0).$$

For the case $\gamma < 0$, we instead get

$$N \leq q^{4\epsilon - (2n - 2(k - 1)) + 1} 10^\epsilon \quad (\gamma < 0).$$

Since $\gamma < 0$ then $5\epsilon < 3n - 4(k - 1)$, so $3\epsilon < \frac{9}{5}n - \frac{12}{5}(k - 1)$, and therefore

$$4\epsilon - (2n - 2(k - 1)) + 1 < \epsilon - (d + 3(k - 1))/5 + 1.$$

That means

$$N < q^{\epsilon - (d + 3(k - 1))/5 + 1} 10^\epsilon \quad (\gamma < 0).$$

As previously described $N/(q - 1)^\epsilon$ then becomes a bound on the probability of decoding failure. The term $(\frac{q}{q-1})^\epsilon$ then appears, but $(\frac{q}{q-1})^\epsilon \leq (\frac{q}{q-1})^q \leq 2^2$. Finally, $10^\epsilon = (q^{-8})^{-\epsilon \log 10 / (8 \log q)}$ and $\log 10 / 8 < 0.29$. \square

Lemma 6.1. *Let $A, B \in \mathbb{F}[x]$ with $\gcd(A, B) = 1$, and $K_1 < K_2 < K_3 \in \mathbb{Z}_+$, as well as $q = |\mathbb{F}|$. Let S denote the set of triples $(f_1, f_2, f_3) \in \mathbb{F}[x]^3$ such that $Bf_2^2 \equiv f_1f_3 \pmod{A}$, while $\deg f_t \leq K_t$ and f_2 is monic. Then*

$$|S| \leq 2^{K_1 + K_3} q^{K_2 + 1 + \max(0, \gamma)},$$

where $\gamma = \max(K_1 + K_3, 2K_2 + \deg B) - \deg A$.

Proof. Consider first $\gamma < 0$ in which case $Bf_2^2 = f_1f_3$. We can choose f_2 in q^{K_2-1} ways. The prime divisors of Bf_2^2 should then be distributed among f_1 and f_3 , which can be done in $2^{K_1+K_3}$ ways. Finally, the leading coefficient of f_1 can be chosen in $q - 1$ ways.

Consider now $\gamma \geq 0$. We choose again first f_2 in one of q^{K_2-1} ways. Then f_1f_3 must be in the set $\{Bf_2^2 + pA \mid p \in \mathbb{F}[x], \deg p \leq \gamma\}$, having cardinality at most $q^{\gamma+1}$. For each of these choices of f_1f_3 , we can again choose f_1 and f_3 in at most $(q - 1)2^{K_1+K_3}$ ways. \square

The bound of Proposition 8 demonstrates a rapid, exponential decrease in the probability of failure as the number of errors decrease away from $\tau_{\text{Pow}}(2,3)$. The bound only becomes non-trivial a few errors below $\tau_{\text{Pow}}(2,3)$, due to the term $0.29\epsilon / \log q - 1/4$ in the exponent. For instance, for a $[32, 9]$ code over $GF(32)$, then $\tau_{\text{Pow}}(2,3) = 13$, but the exponent to q^{-8} in the failure probability bound is only positive for $\epsilon \leq 12$. Such a penalty is not observed in simulations, however, and seems to be an artefact of our proof. For the $[32, 9]$ code, decoding succeeds almost always with 13 errors (see next section). Similarly, for a $[256, 63]$ code over $GF(256)$, the proposition is only non-trivial for $\epsilon < 108$ while in simulations, decoding works almost always up to $\lfloor \tau_{\text{Pow}}(2,3) \rfloor = 112$.

Nevertheless, in an asymptotic and relative sense, Proposition 8 guarantees that decoding up to $\tau_{\text{Pow}}(2, 3)$ almost always succeeds:

Corollary 4. *When $s = 2$ and $\ell = 3$, with $n \rightarrow \infty$ while keeping q/n , k/n and ϵ/n constant, the probability that Power decoding fails goes to 0 when $\epsilon/n < \tau_{\text{Pow}}(2, 3)/n$.*

Proof. Consider the high-error failure probability of Proposition 8:

$$4(q^{-8})^{(\tau_{\text{Pow}}(2,3)-\epsilon)-(0.29\epsilon/\log q-1/4)} \leq 4(q^{-8n})^{\delta-0.29\epsilon/(n \log q)+1/(4n)},$$

where $\delta = \tau_{\text{Pow}}(2, 3)/n - \epsilon/n$. Asymptotically δ approaches some positive constant. The failure probability therefore approaches 0. The low-error probability case is similar. \square

6.1. Failure Behaviour in Relation to List Decoding. It is natural to ask if the failure behaviour of Power decoding is linked to whether or not there are multiple codewords close to the received word, i.e. the list of codewords that e.g. the Guruswami–Sudan algorithm would return. There seems, however, to be no clear relation like this, as we explain below.

Consider that $\mathbf{c} \in \mathcal{C}$ was sent and \mathbf{r} was received. Suppose Power decoding has a decoding radius of τ , and that we have a list decoder of the same decoding radius. Consider that $\mathbf{c}' \in \mathcal{C}$ is another codeword such that either \mathbf{c} or \mathbf{c}' is closest to \mathbf{r} . Then there are the following possibilities:

1. $\text{dist}(\mathbf{c}, \mathbf{r}) = \text{dist}(\mathbf{c}', \mathbf{r}) \leq \tau$
2. $\text{dist}(\mathbf{c}', \mathbf{r}) < \text{dist}(\mathbf{c}, \mathbf{r}) \leq \tau$
3. $\text{dist}(\mathbf{c}, \mathbf{r}) < \text{dist}(\mathbf{c}', \mathbf{r}) \leq \tau$
4. $\text{dist}(\mathbf{c}, \mathbf{r}) \leq \tau < \text{dist}(\mathbf{c}', \mathbf{r})$
5. $\text{dist}(\mathbf{c}', \mathbf{r}) \leq \tau < \text{dist}(\mathbf{c}, \mathbf{r})$
6. $\tau < \text{dist}(\mathbf{c}, \mathbf{r}), \text{dist}(\mathbf{c}', \mathbf{r})$

Clearly, both Power decoding and the list decoder will fail in recovering \mathbf{c} in Item 5 and Item 6. In Items 1–4, the list decoder guarantees to recover \mathbf{c} on a list, though for Item 1–3 that list will have length at least 2.

For Power decoding it is less clear-cut. Firstly, for Items 1 and 2, then Power decoding “fails” according to the definition given at the beginning of Section 6. Indeed, for Items 2, then Power decoding guarantees to return \mathbf{c}' or fail. For Item 1, however, then Algorithm 1 might be lucky and find \mathbf{c} , but in all likelihood $\hat{\mathbf{v}}$ becomes a random linear combination of the solution vectors corresponding to \mathbf{c} and \mathbf{c}' , and Line 4 therefore returns fail. For Items 3 and 4, then Power decoding will probably obtain \mathbf{c} , but it might fail in either case. That is, whether there is only one codeword within radius τ or not, then Power decoding might succeed or it might fail.

That might be surprising at first, so we give examples of these cases. Consider \mathcal{C} to be the $[23, 7]_{GF(23)}$ GRS code with $\alpha_i = i - 1$ and $\beta_i = 1$ for $i = 1, \dots, 23$. For this code, $\tau_{\text{Pow}}(2, 3) = 9$. As an example for Item 3 where Power decoding succeeds, consider the following vectors:

$$\begin{aligned} \mathbf{c}_3 &= (16 \ 15 \ 20 \ 20 \ 3 \ 0 \ 18 \ 0 \ 19 \ 16 \ 2 \ 11 \ 11 \ 3 \ 9 \ 18 \ 5 \ 0 \ 0 \ 0 \ 5 \ 0 \ 16) \in \mathcal{C}, \\ \mathbf{r}_3 &= (16 \ 0 \ 20 \ 20 \ 0 \ 0 \ 18 \ 0 \ 19 \ 0 \ 2 \ 11 \ 0 \ 0 \ 0 \ 5 \ 0 \ 0 \ 0 \ 5 \ 0 \ 0) . \end{aligned}$$

Then $8 = \text{dist}(\mathbf{r}_3, \mathbf{c}_3) < \text{dist}(\mathbf{r}_3, \mathbf{0}) = \tau = 9$. So a list decoder with decoding radius 9 would obtain the list $[\mathbf{0}, \mathbf{c}_3]$. Power decoding returns \mathbf{c}_3 . An example for Item 4 where Power decoding fails, even though there is only one nearby codeword, $\mathbf{0}$, is

$[n, k]_{\mathbb{F}}$	(s, ℓ)	τ_{Pow}	$P_f(\lfloor \tau_{\text{Pow}} \rfloor - 1)$	$P_f(\lfloor \tau_{\text{Pow}} \rfloor)$	$P_f(\lfloor \tau_{\text{Pow}} \rfloor + 1)$	N
$[24, 7]_{GF(25)}$	$(2, 4)$	10	0	6.800×10^{-5}	$1 - 5.8 \times 10^{-5}$	10^6
$[32, 9]_{GF(32)}$	$(2, 3)$	$13\frac{1}{4}$	0	0	$1 - 4.20 \times 10^{-4}$	10^6
$[22, 3]_{GF(23)}$	$(6, 18)$	14	4.350×10^{-4}	1.414×10^{-2}	1	10^6
$[64, 29]_{GF(64)}$	$(4, 5)$	19	0	0	1	10^5
$[68, 31]_{GF(71)}$	$(3, 4)$	20	0	0	1	10^6
$[125, 51]_{GF(125)}$	$(4, 6)$	42	0	0	1	10^5
$[256, 63]_{GF(256)}$	$(2, 4)$	$116\frac{2}{5}$	0	0	$1 - 3.00 \times 10^{-4}$	10^5

TABLE 1. Simulation results. $P_f(\tau)$ denotes the observed probability of decoding failure (no result or wrong result) with random errors of weight exactly τ .

the following received word:

$$\mathbf{r}_4 = (0\ 2\ 9\ 1\ 0\ 0\ 6\ 0\ 0\ 0\ 5\ 0\ 0\ 0\ 0\ 0\ 4\ 8\ 15\ 0\ 0\ 12) .$$

That example was found by random generation of error vectors of weight 9, after roughly 47 000 successful decoding trials. As an aside, the failure probability bound of Proposition 8 yields a number greater than 1 for this code, parameters and number of errors.

7. Simulation Results. The proposed decoding algorithm has been implemented in Sage v6.9 [39], and is available for download at <http://jsrn.dk/code-for-articles>. The implementation uses the punctured module described in Section 4.1 and computes the weak Popov form using the Mulders–Storjohann algorithm [23]. The asymptotic complexity of the implementation is therefore $O(\ell^3 s^2 n^2)$.

To evaluate the failure probability, we have selected a range of code and decoding parameters and run the algorithm for a large number of random errors. More precisely, for each set of parameters, and each decoding radius τ , we have created N random errors of weight exactly τ and attempted to decode a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$ for some randomly chosen \mathbf{c} (though, of course, Proposition 6 implies that shifting by \mathbf{c} makes no difference). We have limited the decoding radii used to being $\lfloor \tau_{\text{Pow}}(s, \ell) \rfloor + \{-1, 0, 1\}$. N was either 10^5 or 10^6 for a given parameter set. The results are listed as Table 1.

As is evident, $\tau_{\text{Pow}}(s, \ell)$ very clearly describes the number of errors we can rely on correcting: the probability of failing appears to decay exponentially with $\tau_{\text{Pow}}(s, \ell) - \epsilon$, as we might expect if extrapolating from the bound of Proposition 8. In fact, the failure probability is so low that it is difficult to observe failing cases for randomly selected errors.

The case having the highest failure rate is the very low-rate code $[22, 3]_{GF(23)}$. For such a low-rate code, $\tau_{\text{Pow}}(s, \ell)$ is quite close to the minimum distance, and there is a significant probability that a random error will yield a received codeword which is closer to another received word. In this case, Power decoding always fails. We performed another simulation for this code with 10^4 random errors of weight exactly 14 and decoding using the Guruswami–Sudan list decoder. This simulation had a 1.23×10^{-2} probability that another codeword was as close or closer to the sent codeword. Thus most of the Power decoding failures stem from this.

8. Re-Encoding. “Re-Encoding” is a simple technique invented by Kötter and Vardy, originally for reducing the complexity of the interpolation step in the Guruswami–Sudan algorithm [18]. It is especially powerful when using different multiplicities at each point, such as in the Kötter–Vardy soft-decision decoding version of Guruswami–Sudan [17]. For the regular Guruswami–Sudan, and in usual asymptotic analysis where k/n is considered a constant, re-encoding does not change the asymptotic cost; however, it can have a significant practical impact on the running time, especially for higher-rate codes. We will now show that the re-encoding transformation easily applies to Power decoding as well.

Consider that $\hat{\mathbf{r}}$ is the received word. Using Lagrange interpolation, we can easily compute the unique $\hat{\mathbf{c}} = \text{ev}(\hat{f}) \in \mathcal{C}$ such that $\hat{\mathbf{c}}$ and $\hat{\mathbf{r}}$ coincide on the first k positions. Clearly, decoding $\mathbf{r} = \hat{\mathbf{r}} - \hat{\mathbf{c}}$ immediately gives a decoding of $\hat{\mathbf{r}}$. The idea of re-encoding is that the leading k zeroes of the resulting \mathbf{r} might be utilised in the decoding procedure to reduce the computation cost of decoding \mathbf{r} .

Assume therefore for this section that \mathbf{r} is the received word after re-encoding and therefore has k leading zeroes. That means $\hat{G} \mid R$ where $\hat{G} = \prod_{i=1}^k (x - \alpha_i)$. Consider the linearised key equations of Problem 3. Each of them are now divisible by $\hat{G}^{\min(s,t)}$, and so we obtain:

$$\begin{aligned} 1a) \quad \psi_t / \hat{G}^t &= \sum_{i=0}^t \lambda_i \cdot \left(\binom{t}{i} R^{t-i} G^i \hat{G}^{-t} \right), & \text{for } t = 1, \dots, s-1 \\ 1b) \quad \psi_t / \hat{G}^s &\equiv \sum_{i=0}^{s-1} \lambda_i \cdot \left(\binom{t}{i} R^{t-i} G^i \hat{G}^{-s} \right) \pmod{(G/\hat{G})^s}, & \text{for } t = s, \dots, \ell. \end{aligned}$$

The elements $\dot{\psi}_t \triangleq \psi_t / \hat{G}^{\min(s,t)}$ and $R^{t-i} G^i \hat{G}^{-\min(s,t)}$ are all polynomials, but of much lower degree than before. Thus, we can solve for λ_i and $\dot{\psi}_t$ directly, being a system of fewer variables. The degree restriction on $\dot{\psi}_t$ becomes

$$\deg \lambda_0 + t(k-1) - \min(s,t)k \geq \deg \dot{\psi}_t.$$

Note that re-encoding will not change the failure behaviour: by Proposition 6, the re-encoded equations before dividing through by $G^{\min(s,t)}$ will have solutions in one-to-one correspondence with those of the original equations. After dividing through by $G^{\min(s,t)}$, this is still true.

8.1. Solving the Re-Encoded Equations. We solve these key equations exactly in the same way as before: construct an $\mathbb{F}[x]$ -matrix whose row space contains all solutions to the congruences $(\lambda_0, \dots, \lambda_{s-1}, \dot{\psi}_1, \dots, \dot{\psi}_\ell)$, and we find the sought minimal solution as a lowest weighted-degree vector in this row space with λ_0 monic. We then hope that this vector equals $\dot{\mathbf{v}} = (\Lambda^s, \dots, \Lambda \Omega^{s-1}, \Lambda^s f / \hat{G}, \dots, \Lambda^s f^s / \hat{G}^s, \dots, \Lambda^\ell f^\ell / \hat{G}^s)$, where f is the information polynomial after re-encoding has been applied.

In the matrix M of Proposition 1, this is reflected as dividing by $\hat{G}^{\min(s,t)}$ through each column for $t = s+1, \dots, \ell+s$, obtaining \tilde{M} of reduced degree.

To find a minimal *weighted*-degree vector in the row space of \tilde{M} , we again multiply on the right by an appropriately selected diagonal matrix \tilde{D} containing only powers of x . Explicitly, we introduce non-negative variables $\dot{\mu}_i, \dot{\eta}_t \in \mathbb{N}_0$:

$$\dot{\mu}_0 = 1 + \ell(k-1) - sk \dot{\mu}_i = i + \ell(k-1) - sk, \quad i > 0 \quad \dot{\eta}_t = (\ell-t)(k-1) - (s - \min(s,t))k, \quad (13)$$

Then we select $\tilde{D} = \text{diag}(x^{\dot{\mu}_0}, \dots, x^{\dot{\mu}_{s-1}}, x^{\dot{\eta}_1}, \dots, x^{\dot{\eta}_\ell})$.

We then compute \hat{B} unimodular equivalent to \hat{M} and such that $\hat{B}\hat{D}$ is in weak Popov form. If we are fortunate, $\hat{v}\hat{D}$ will be the lowest-degree row in $\hat{B}\hat{D}$ having leading position on the first position. From this, we get Λ^s and $\Lambda f/\hat{G}^s$, and so we can calculate f . Finally, the original information related to the original received word \mathbf{r} is then $f + \hat{f}$.

According to Proposition 4 the complexity of finding such $\hat{B}\hat{D}$ depends quasi-linearly on $\deg(\hat{M}\hat{D})$:

$$\deg(\hat{M}\hat{D}) \leq s(n - k) + (\ell - s)k \leq 2s(n - k) ,$$

where the last inequality comes from the general assumption that $\ell k < sn$ that we mentioned in Section 4. Thus, the complexity of finding \hat{B} becomes $O^\sim((\ell + s)^\omega s(n - k))$. In standard asymptotic analyses, we assume $k \in \Theta(n)$, in which case this equals $O^\sim((\ell + s)^\omega sn)$. However, in practice, the re-encoding technique should give a noticeable speedup. As a last remark, note that the puncturing proposed in Section 4.1 applies equally well to the re-encoded module, yielding the complexity $O^\sim(\ell^\omega s(n - k))$.

9. Syndrome Key Equations. As described in Section 2, the first key equation decoding algorithm was based on the notion of syndrome polynomial [6], and similarly, Power decoding without multiplicities was first described using a similar list of key equations [36]. The key equations of Theorem 3.1 can similarly be rewritten to be based on syndrome polynomials, which we will show in this section. As is usual for syndrome-formulated key equations, we will assume that 0 is not used as an evaluation point. Therefore $x \nmid G$. Furthermore, due to a non-essential technicality, we will assume $s < n$. If this did not hold, the following analysis of parameters would be slightly more complicated but not impossible.

Recall the reversal operator $\text{rev}(p, d)$ which we defined in Section 2.2. Define for a given value of the multiplicity s the following variants of the powered Lagrange interpolant R as well as a generalised notion of syndrome:

$$R^{(i,t)} \triangleq R^{t-i} \bmod G^{s-i} \qquad S^{(i,t)} = \frac{\text{rev}(R^{(i,t)})}{\text{rev}(G)^{s-i}} .$$

Note the degree that the reversal-operator on $\text{rev}(R^{(i,t)})$ uses: if $t - i \leq s - i$ then $R^{(i,t)} = R^{t-i}$ so the degree upper bound is $(t - i)(n - 1)$. If $t - i > s - i$ then $\deg R^{t-i} > \deg G^{s-i}$ since we have assumed $s < n$, and therefore $\deg R^{(i,t)} \leq (s - i)n - 1$.

If $s = 1$ then $S^{(1,1)}$ equals the classical syndrome polynomial S which we used in Section 2.2, and $S^{(1,t)}$ equals the syndromes $S^{(t)}$ discussed in Section 2.3. The syndromes also appear (with a slightly different definition) in the Interpolation key equations for Guruswami–Sudan by Gentner et al. [46].

We can then formulate the—markedly more involved—syndrome variant of Theorem 3.1:

Theorem 9.1. *For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then there exist $g_t \in \mathbb{F}[x]$ for $t = s, \dots, \ell$ such that*

$$\begin{aligned} \sum_{i=0}^t \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} S^{(i,t)} \right) &\equiv 0 \pmod{x^{\varrho_t}} && \text{for } t = 1, \dots, s-1, \\ \sum_{i=0}^t \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} S^{(i,t)} x^{\iota_{i,t}} \right) &\equiv g_t \pmod{x^{\varrho_t}} && \text{for } t = s, \dots, \ell, \end{aligned}$$

where

$$\begin{aligned} \deg g_t &\leq \begin{cases} \epsilon s - s & \text{if } t = s \\ \epsilon s - 1 & \text{if } t > s \end{cases} \\ \varrho_t &= \begin{cases} t(n-k) & \text{if } t \leq s \\ sn - t(k-1) - 1 & \text{otherwise} \end{cases} \\ \iota_{i,t} &= \begin{cases} 0 & \text{if } t = s \\ i & \text{if } t > s \end{cases}. \end{aligned}$$

Proof. We need to distinguish between two cases: $t < s$ and $t \geq s$. Assume first $t < s$. Since $R^{(i,t)} = R^{t-i}$, Theorem 3.1 gives us

$$\begin{aligned} \sum_{i=0}^t (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{(i,t)} G^i \right) &= \Lambda^s f^t && \Longleftrightarrow \\ \text{rev} \left(\sum_{i=0}^t (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{(i,t)} G^i \right), \epsilon s + t(n-1) \right) &= \text{rev}(\Lambda^s f^t, \epsilon s + t(n-1)), \end{aligned}$$

where $\epsilon s + t(n-1)$ arise from counting the degree upper bound on the left-hand side. Every term in the sum has the same degree bound, so we get

$$\begin{aligned} \sum_{i=0}^t \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} \text{rev}(R^{(i,t)}) \text{rev}(G)^i \right) &= \text{rev}(\Lambda^s f^t) x^{t(n-k)} && \Longrightarrow \\ \sum_{i=0}^t \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} \text{rev}(R^{(i,t)}) \text{rev}(G)^i \right) &\equiv 0 \pmod{x^{t(n-k)}} && \Longleftrightarrow \\ \sum_{i=0}^t \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} S^{(i,t)} \right) &\equiv 0 \pmod{x^{t(n-k)}}, \end{aligned}$$

where the last line follows from $\text{rev}(G)^s$ being invertible modulo $x^{t(n-k)}$. This concludes the case $t < s$.

For the case $t \geq s$, we proceed similarly. In the congruence of Theorem 3.1, we can readily replace $R^{t-i} G^i$ with $R^{(i,t)} G^i$ modulo G^s . This gives us:

$$\begin{aligned} \Lambda^s f^t &\equiv \sum_{i=0}^{s-1} (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{(i,t)} G^i \right) \pmod{G^s} && \Longrightarrow \\ \Lambda^s f^t + \text{rev}(g_t) G^s &= \sum_{i=0}^{s-1} (\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} R^{(i,t)} G^i \right), \end{aligned}$$

for some $\text{rev}(g_t) \in \mathbb{F}[x]$. The degree of the right-hand side is bounded as

$$\max_i \{ (s-i)\epsilon + i(\epsilon-1) + \deg R^{(i,t)} + in \} \leq \begin{cases} s\epsilon + s(n-1) & \text{if } t = s \\ s\epsilon + sn - 1 & \text{if } t > s \end{cases}.$$

This immediately bounds $\deg g_t$ as the theorem states. Note that the above equals $\varrho_t + s\epsilon + t(k-1)$ in all cases. We can now reverse the equation as in the previous case. When $t > s$ then the degree bound on the summands are not all the same, so

we must add powers of x in the reversed expression:

$$\begin{aligned}
\text{rev}(\Lambda^s f^t) x^{2t} + g_t \text{rev}(G)^s &= \sum_{i=0}^{s-1} \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} \text{rev}(R^{(i,t)}) \text{rev}(G)^i x^{\iota_{i,t}} \right) \implies \\
g_t \text{rev}(G)^s &\equiv \sum_{i=0}^{s-1} \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} \text{rev}(R^{(i,t)}) \text{rev}(G)^i x^{\iota_{i,t}} \right) \pmod{x^{2t}} \iff \\
g_t &\equiv \sum_{i=0}^{s-1} \text{rev}(\Lambda^{s-i} \Omega^i) \left(\binom{t}{i} S^{(i,t)} x^{\iota_{i,t}} \right) \pmod{x^{2t}}.
\end{aligned}$$

□

Remark 3. Just as we remarked that the key equations of Theorem 3.1 resemble certain characterisations of Interpolation polynomials in Guruswami–Sudan, so does the above syndrome formulation resemble the syndrome Interpolation key equations of [46]. Again, the deeper relation between the error locator approach and the Guruswami–Sudan is unclear.

Theorem 9.1 leads to a decoding algorithm in the very same way as Theorem 3.1. We could call these algorithms “Power syndromes” and “Power Gao” respectively. We have the following important remark:

Corollary 5. *Decoding using Power Gao succeeds if and only if decoding using Power syndromes succeeds.*

Proof. This follows easily by the same transformation as in the proof of Theorem 9.1: a solution to the linearised key equations of Power Gao induces a solution to the linearised key equations of Power syndromes, and vice versa. □

Thus the two decoding algorithms have exactly the same decoding performance.

9.1. Solving the Syndrome Equations. To use the key equations of Theorem 9.1 for decoding, we again proceed in a manner similar to that of Section 4: we linearise the problem by forgetting the algebraic connection between the $\text{rev}(\Lambda^{s-i} \Omega^i)$ and g_t . The problem then becomes finding $\mathbf{v} = (\text{rev}(\Lambda^s), \text{rev}(\Lambda^{s-1} \Omega), \dots, \text{rev}(\Lambda \Omega^{s-1}), 0, \dots, 0, g_s, \dots, g_\ell)$ as the, hopefully, lowest weighted-degree vector in the row space of an explicit $\mathbb{F}[x]$ -matrix, M_{Syn} : a problem we can solve by applying lattice basis reduction techniques. We will not go through all the details as in Section 4 since the technique is so similar.

$M_{\text{Syn}} \in \mathbb{F}[x]^{(s+\ell) \times (s+\ell)}$ becomes

$$M_{\text{Syn}} = \left[\begin{array}{c|c} \mathbf{I}_{s \times s} & N_{\text{Syn}} \\ \hline \mathbf{0}_{\ell \times s} & \text{diag}(x^{\rho_1}, \dots, x^{\rho_\ell}) \end{array} \right],$$

where $N_{\text{Syn}} \in \mathbb{F}[x]^{s \times \ell}$ is the matrix whose (i, t) th entry is

$$N_{\text{Syn}}[i, t] = \binom{t}{i-1} S^{(i-1,t)} x^{\iota_{i-1,t}}, \quad i = 1, \dots, s \text{ and } t = 1, \dots, \ell.$$

To find a low weighted-degree vector in the row space of M_{Syn} , we again transform $M_{\text{Syn}} D_{\text{Syn}}$ into weak Popov form, where D_{Syn} is an appropriately chosen diagonal matrix, containing only powers of x . D_{Syn} should be chosen such that the degrees of the entries of the sought solution $\mathbf{v} D_{\text{Syn}}$ all have roughly the same degrees while

retaining $\text{LP}(\mathbf{v}D_{\text{Syn}}) = 1$. To handle the first 0 entries, we simply use a large weight. Explicitly, we can select:

$$D_{\text{Syn}} = \text{diag}(x^1, \quad x^1, x^2, \dots, x^{s-1}, \quad \underbrace{x^{s\tau_{\text{Pow}}(s,\ell)}, \dots, x^{s\tau_{\text{Pow}}(s,\ell)}}_{s-1 \text{ times}}, \quad x^s, \quad \underbrace{x^1, \dots, x^1}_{\ell-s \text{ times}}).$$

We then compute B_{Syn} unimodular equivalent to M_{Syn} and such that $B_{\text{Syn}}D_{\text{Syn}}$ is in weak Popov form. If we are fortunate, $\mathbf{v}D_{\text{Syn}}$ will be the lowest-degree row in $B_{\text{Syn}}D_{\text{Syn}}$ having leading position on the first position, up to an \mathbb{F} -multiple. From this, we immediately get Λ and Ω , and so we can calculate f using e.g. Lemma 2.1.

Similarly to the re-encoding case in Section 8, the cost of computing B_{Syn} seems to be lower than minimising MD directly, but not asymptotically so. In particular, the asymptotic complexity has quasi-linear dependence on $\deg(M_{\text{Syn}}D_{\text{Syn}}) \in O(s(n-k))$. This is similar to when using the re-encoding technique of Section 8. However, the puncturing of the module done in Section 4.1 cannot be replicated for Power syndrome since that would result in an over-defined basis of the resulting punctured module (the punctured matrix would have more rows than columns). This in turn would drastically reduce the decoding radius. Thus, the Power syndrome variant must reduce an $(s+\ell)^2$ matrix, while Power Gao, with or without re-encoding, reduces an $(\ell+1)^2$ matrix. On the other hand, it might be possible that the $s-1$ zero-remainder congruences of Power syndrome could be handled in a faster manner than described here. Without a much finer analysis and concrete choices of algorithms for computing the weak Popov form, we cannot conclude which algorithm will be fastest and by how much.

10. Conclusion. We demonstrated how the Power decoding technique for Reed–Solomon codes can be augmented with a new parameter—the multiplicity—to attain the Johnson decoding radius [15]. The resulting decoder is, as the original Power decoding algorithm of Schmidt, Sidorenko and Bossert [36], a partial decoder which fails for a few received words within its decoding radius. We showed how one can efficiently solve the resulting key equations using lattice basis reduction techniques to obtain a complexity close to the fastest realisation of the Guruswami–Sudan or Wu list decoding algorithms [45], namely $O^{\sim}(\ell^{\omega} sn)$.

The proposed decoder has applications as a simpler alternative to the Guruswami–Sudan algorithm—especially for hardware implementations—due to its simple one-step shift-register type structure. In particular, for medium rate codes, Power decoding with a low multiplicity of 2 or 3 would not be much more complicated to implement than half-the-minimum distance decoding while still offering a significant improvement in decoding performance. For such parameters, the root-finding step of the Guruswami–Sudan algorithm would have a significant circuit area and latency.

The exact failure behaviour of the decoding method remains largely open. For $s = 1$, i.e. the original Power decoding, the failure probability has previously been bounded only for $\ell = 2, 3$. The case $s > 1$ seems no easier to analyse: Proposition 6 simplifies the equations one needs to analyse, and this was instrumental in the case for which we were able to bound the failure probability: $(s, \ell) = (2, 3)$. For these parameters, the decoding radius improves upon the case $s = 1$ whenever the rate is within $]1/6; 1/2[$. The remaining open questions on the failure probability were counterbalanced by simulations on a range of codes: this demonstrates a failure probability which seems to decay exponentially as the number of errors is reduced.

Power decoding has already been applied for various related codes, e.g. improved decoding of Interleaved RS codes [44] and Complex RS codes [22]. It is clear that the proposed addition of multiplicities can aid those applications as well, and this is an interesting avenue of future work. Another interesting question is to extend Power decoding to soft-decision decoding, similar to Kötter–Vardy’s variant of the Guruswami–Sudan algorithm [17].

We also discussed two variants of the decoding method which reduces the cost in practice: re-encoding and a syndrome formulation. Either method roughly replaces the complexity dependency on n with $n - k$. More detailed analysis, and concrete choices of basis reduction algorithms is necessary to determine which one is fastest in practice.

11. Acknowledgements. The author would like to thank Vladimir Sidorenko, Martin Bossert and Daniel Augot for discussions on Power decoding and this paper. The author gratefully acknowledges the support of the Digiteo foundation, project [IdealCodes](#) while he was with Inria, and also, while the author was with Ulm University, the support of the German Research Council “Deutsche Forschungsgemeinschaft” (DFG) under grant BO 867/22-1.

References

- [1] A. Ahmed, R. Koetter, and N. R. Shanbhag. VLSI Architectures for Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Trans. Inf. Theory*, 57(2):648–667, Feb. 2011.
- [2] M. Alekhnovich. Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed–Solomon Codes. *IEEE Trans. Inf. Theory*, 51(7):2257–2265, July 2005.
- [3] G. Baker and P. Graves-Morris. *Padé approximants*, volume 59. Cambridge Univ. Press, 1996.
- [4] P. Beelen and K. Brander. Key equations for list decoding of Reed–Solomon codes and how to solve them. *J. Symb. Comp.*, 45(7):773–786, 2010.
- [5] P. Beelen, T. Høholdt, J. S. R. Nielsen, and Y. Wu. On Rational Interpolation-Based List-Decoding and List-Decoding Binary Goppa Codes. *IEEE Trans. Inf. Theory*, 59(6):3269–3281, June 2013.
- [6] E. R. Berlekamp. *Algebraic Coding Theory*. Aegean Park Press, 1968.
- [7] M. Chowdhury, C.-P. Jeannerod, V. Neiger, E. Schost, and G. Villard. Faster Algorithms for Multivariate Interpolation With Multiplicities and Simultaneous Polynomial Approximations. *IEEE Trans. Inf. Theory*, 61(5):2370–2387, May 2015.
- [8] H. Cohn and N. Heninger. Ideal forms of Coppersmith’s theorem and Guruswami–Sudan list decoding. *arXiv*, 1008.1284, 2010.
- [9] P. Fitzpatrick. On the Key Equation. *IEEE Trans. Inf. Theory*, 41(5):1290–1302, 1995.
- [10] S. Gao. A New Algorithm for Decoding Reed-Solomon Codes. In *Communications, Information and Network Security*, number 712 in S. Eng. and Comp. Sc., pages 55–68. Springer, Jan. 2003.
- [11] P. Giorgi, C. Jeannerod, and G. Villard. On the Complexity of Polynomial Matrix Computations. In *Proc. of ISSAC*, pages 135–142, 2003.
- [12] Gottfried Ungerböck. Improved burst-error correction by joint decoding of interleaved RS codes. In *Munich Workshop on Coding and Modulation*, July 2015.
- [13] S. Gupta, S. Sarkar, A. Storjohann, and J. Valeriote. Triangular -basis decompositions and derandomization of linear algebra algorithms over. *J. Symb. Comp.*, 47(4):422–453, Apr. 2012.
- [14] V. Guruswami and M. Sudan. Improved Decoding of Reed–Solomon Codes and Algebraic-Geometric Codes. *IEEE Trans. Inf. Theory*, 45(6):1757–1767, 1999.
- [15] S. M. Johnson. A New Upper Bound for Error-Correcting Codes. *IEEE Trans. Inf. Theory*, 46:203–207, 1962.
- [16] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.
- [17] R. Kötter and A. Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Trans. Inf. Theory*, 49(11):2809–2825, 2003.
- [18] R. Kötter and A. Vardy. A Complexity Reducing Transformation in Algebraic List Decoding of Reed–Solomon Codes. In *Proc. of IEEE ITW*, 2003.

- [19] K. Lee and M. E. O'Sullivan. List Decoding of Reed–Solomon Codes from a Gröbner Basis Perspective. *J. Symb. Comp.*, 43(9):645 – 658, 2008.
- [20] K. Lee and M. E. O'Sullivan. List decoding of Hermitian codes using Gröbner bases. *J. Symb. Comp.*, 44(12):1662–1675, 2009.
- [21] W. Li, J. S. R. Nielsen, and V. R. Sidorenko. On Decoding of Interleaved Chinese Remainder Codes. In *Proc. of MTNS*, 2014. Extended abstract.
- [22] M. Mohamed, S. Rizkalla, H. Zoerlein, and M. Bossert. Deterministic Compressed Sensing with Power Decoding for Complex Reed-Solomon Codes. In *Proc. of SCC*, 2015.
- [23] T. Mulders and A. Storjohann. On Lattice Reduction for Polynomial Matrices. *J. Symb. Comp.*, 35(4):377–401, 2003.
- [24] J. Nielsen and P. Beelen. Sub-Quadratic Decoding of One-Point Hermitian Codes. *IEEE Trans. Inf. Theory*, 61(6):3225–3240, June 2015.
- [25] J. S. R. Nielsen. Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation. In *Proc. of IEEE ISIT*, pages 882–886, 2013.
- [26] J. S. R. Nielsen. *List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2013.
- [27] J. S. R. Nielsen. Power Decoding of Reed–Solomon Codes Revisited. In *Proc. of ICMCTA*, Sept. 2014.
- [28] J. S. R. Nielsen. Power Decoding of Reed–Solomon Up to the Johnson Radius. In *Proc. of ACCT*, Sept. 2014.
- [29] J. S. R. Nielsen and A. Storjohann. Algorithms for Simultaneous Padé Approximations. In *Proc. of ISSAC*, Feb. 2016. Submitted.
- [30] Z. Olesh and A. Storjohann. The vector rational function reconstruction problem. In *Proc. of WWC*, pages 137–149, 2006.
- [31] R. Roth. *Introduction to Coding Theory*. Cambridge Univ. Press, 2006.
- [32] R. Roth and G. Ruckenstein. Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. *IEEE Trans. Inf. Theory*, 46(1):246 –257, 2000.
- [33] S. Sarkar and A. Storjohann. Normalization of Row Reduced Matrices. In *Proc. of ISSAC*, pages 297–304. ACM, 2011.
- [34] G. Schmidt, V. Sidorenko, and M. Bossert. Decoding Reed-Solomon Codes Beyond Half the Minimum Distance Using Shift-Register Synthesis. In *Proc. of IEEE ISIT*, pages 459–463, 2006.
- [35] G. Schmidt, V. Sidorenko, and M. Bossert. Collaborative Decoding of Interleaved Reed–Solomon Codes and Concatenated Code Designs. *IEEE Trans. Inf. Theory*, 55(7):2991–3012, 2009.
- [36] G. Schmidt, V. Sidorenko, and M. Bossert. Syndrome Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis. *IEEE Trans. Inf. Theory*, 56(10):5245–5252, 2010.
- [37] V. Sidorenko and M. Bossert. Fast skew-feedback shift-register synthesis. *Designs, Codes and Cryptography*, 70(1-2):55–67, Jan. 2014.
- [38] V. Sidorenko and G. Schmidt. A Linear Algebraic Approach to Multisequence Shift-Register Synthesis. *Problems of Information Transmission*, 47(2):149–165, 2011.
- [39] W. A. Stein et al. SageMath Software. <http://www.sagemath.org>.
- [40] M. Sudan. Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.
- [41] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A Method for Solving Key Equation for Decoding Goppa Codes. *Information and Control*, 27(1):87–99, 1975.
- [42] P. Trifonov and M. Lee. Efficient Interpolation in the Wu List Decoding Algorithm. *IEEE Trans. Inf. Theory*, 58(9):5963–5971, 2012.
- [43] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, 3rd edition, 2012.
- [44] A. Wachter-Zeh, A. Zeh, and M. Bossert. Decoding interleaved Reed–Solomon codes beyond their joint error-correcting capability. *Designs, Codes and Cryptography*, 71(2):261–281, July 2012.
- [45] Y. Wu. New List Decoding Algorithms for Reed-Solomon and BCH Codes. *IEEE Trans. Inf. Theory*, 54(8):3611–3630, 2008.
- [46] A. Zeh, C. Gentner, and D. Augot. An Interpolation Procedure for List Decoding Reed-Solomon Codes Based on Generalized Key Equations. *IEEE Trans. Inf. Theory*, 57(9):5946–5959, 2011.

- [47] A. Zeh, A. Wachter, and M. Bossert. Unambiguous Decoding of Generalized Reed–Solomon Codes Beyond Half the Minimum Distance. In *Proc. of IZS*, 2012.
 - [48] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symb. Comp.*, 47(7):793–819, July 2012.
- E-mail address:* jsrn@jsrn.dk